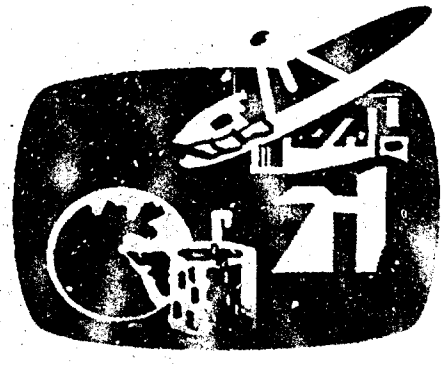
USAISEC
US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES
(AIRMICS)

ANSWER PHASE I - FINAL REPORT

(ASQBG-I-89-027)

July, 1989



AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704--188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION / AVAILABILITY OF REPORT N/A
2b. DECLASSIFICATION / DOUWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ASQBG-I-89-027	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION AIRMICS 6b. OFFICE SYMBOL (if applicable) ASQBG - I	7a. NAME OF MONITORING ORGANIZATION N/A
6c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	7b. ADDRESS (City, State, and Zip Code) N/A
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS 8b. OFFICE SYMBOL (if applicable) ASQBG - I	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N/A
8c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 62783A PROJECT NO.: DY10 TASK NO.: 04-01 WORK UNIT ACCESSION NO.:

11. TITLE (Include Security Classification)

Army's Nonprogrammer System for Working Encyclopedia Requests (ANSWER) (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)

Dr. Karen Ryan, Cho-Li Hou, Datta Shetti

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM _____ TO _____	June 15, 1989	51

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Data Encyclopedia, Very Large Database, Distributed Query Processing, Metadata

19. ABSTRACT (Continue on reverse if necessary and identity by block number)

This report describes research efforts into the access of distributed heterogeneous databases through an encyclopedia facility. Specifically, several data management tools that have been prototyped to date are described to include database registration, schema integration, browsing, and an Information Resource Dictionary System (IRDS) repository. Plans for future efforts to include AI techniques for data management, security, distributed query formulation, and distributed query processing are also discussed.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED / UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL CPT Joseph J. Nealon	22b. TELEPHONE (Include Area Code) (404) 894-3110 22c. OFFICE SYMBOL ASQBG - I

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

This research was performed under contract number DAKF11-88-C-0024 for the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the Army's Information Systems Engineering Command (ISEC). This report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

s/ Glenn Racine
Glenn Racine, Chief
Computer and Information
Systems Division

s/ John R. Mitchell
John R. Mitchell
Director
AIRMICS

Army's Nonprogramming System for Working Encyclopedia Requests (ANSWER)

Final Report

DAKF11-88-C-0024

Prepared for:

Honeywell Federal Systems Inc.

Prepared by:

Karen Ryan
Cho-Li Hou
Datta Shetti

June 15, 1989

Honeywell

Corporate Systems Development Division
1000 Boone Avenue North
Golden Valley, MN 55427

G89082

Table of Contents

List of Figures

List of Figures (Concluded)

Section 1
Introduction

This document is the final report for Phase I of the Army's Non-programmer System for Working Encyclopedia Requests (ANSWER) program. It covers the work accomplished during the precontract award and in the 12 months of Phase I of the program, ranging from May 17, 1988 to May 16, 1989. A detailed discussion of the work performed during the first six months of the contract may be found in the Interim Technical Report. The current report focuses on a description of the software deliverables for Phase I and their relationship to the overall ANSWER program goals.

1.1 Background

The Army is seeking to improve the usefulness and effectiveness of its information systems and to improve user access to those systems through increasing the interoperability, integration and synchronization of the systems. Maximizing the use of these systems requires that users:

- Have access to large amounts of data,
- Understand the logical and physical location of the data,
- Understand the system-specific procedures required to access the information,
- Understand how to combine data values retrieved to produce an answer to the original query.

These requirements exist in any situation where a user is faced with using data from large heterogeneous information systems, most probably developed and maintained independently.

The Army is taking steps to facilitate access to information in such a large, heterogeneous and distributed environment by:

- Defining the HQDA model,
- Defining an Army information engineering methodology,
- Instituting the Army Data Management and Standards Program (AR 25-9),
- Instituting the Army data dictionary efforts.

What is required now is a comprehensive solution that brings these efforts together.

1.2 Approach

ANSWER addresses data access problems by providing a set of tools to aid data administrators and end users in managing and accessing the data in a manner consistent with the Army's efforts. Specifically, the ANSWER program is intended to provide tools to:

- Create and store schema information and Army Data Dictionary (ADD) information,
- Access and manipulate the schema and ADD information,
- Formulate and execute requests against Army databases.

Since the ANSWER program is a research program, tools produced as part of the ANSWER program will incorporate advanced state-of-the-art concepts and techniques; however these tools will demonstrate feasibility to provide a basis for experimentation and evaluation.

The intent of the ANSWER program is to provide the Army with tools well suited to short-term and long-term Army needs. As part of this effort, we have identified several useful applications for our tools within existing Army organizations. These organizations will use and evaluate the ANSWER Phase I deliverables as part of their current efforts. We intend to continue to make every effort to be flexible within the constraints of the contract, providing useful tools for the Army's data management needs.

1.3 Program Phase Summary

The ANSWER program is four-phase, 36-month program. Phase I has been a 12-month effort. Phase II is a six-month effort refining and expanding the results of Phase I. Phase III is another 12-month effort while Phase IV is defined as another refinement of the final ANSWER deliverables.

The details of the four phases are discussed below.

1.3.1 Phase I

Phase I consists of the following tasks:

- ***Requirements analysis***—Define the functional requirements for ANSWER.
- ***System design***—Define the information and functional architectures for ANSWER.

- ***Representation model***—Choose an appropriate representation schema to describe ANSWER objects

- ***Implementation environment***—Choose a suitable environment in which to implement ANSWER

- ***Encyclopedia***—Design and implement an encyclopedia to store and organize ANSWER information

- ***Automated database registration***—Design and implement a schema integration facility and other tools to aid in the integration of database schemas into the ANSWER system.

- ***Browser***—Design and implement a browsing capability for the ANSWER information architecture.

- ***User interface***—Design a simple user interface that will allow the user to access and manipulate ANSWER tools.

- ***Demonstration and training***.

Phase I is now completed. The data administrator or end user can:

- Store and access database schemas from the Information Resource Directory System (IRDS),

- Invoke the schema integration tool and write the results back to IRDS,

- Create new standard data element associations with schema attributes,

- Graphically browse the ANSWER information architecture and create new ADD associations using the Browser.

1.3.2 Phase II

Phase II consists of the following tasks:

- ***Database registration automation***—In this task we will design modifications to the schema integrator tool. The modifications will be based on feedback received from a target user group using the schema integrator developed in Phase I.We will demonstrate the modified schema integrator at the end of Phase II.

- ***Implementation of AI techniques***—In this task we will investigate database registration tools other than the schema integrator that are applicable to the problem of database registration. Such tools might include data element standardization tools and flat file system

conversion tools. We will investigate possibilities for these other tools in Phase II in cooperation with Army agencies identified with AIRMICS assistance.

- ***Demonstration and training***.

- ***User interface***—In this task we will implement an X-Windows-based user interface that allows the user to invoke the ANSWER tools, including the schema integrator tool, the Browser, IRDS and other tools to be defined as part of the AI techniques task.

- ***Browsing***—In this task we will design possible modifications to the browsing tool developed and delivered as part of Phase I. The modifications will be based on feedback received from a target user group using the software deliverables from Phase I.

1.3.3 Phase III

The tasks for Phase III include:

- ***Implement Browser enhancements***—In this task we will implement enhancements to the Browser. These enhancements were identified in Phase II. We will also investigate issues associated with implementing the Browser on a large, realistic model.

- ***Query formulation implementation***—In this task we will analyze approaches for query formulation, make recommendations for implementing query formulation and review these recommendations with the Army representatives to select appropriate approaches. Approaches to be analyzed include syntax diagram display, SQL syntax error detection, draft SQL execution, intelligent user interfaces, automatic formulation of SQL requests from Browser examples, natural language paraphrases of SQL requests and natural language queries translated into SQL. We will design and implement (as necessary) query formulation tools that are selected.

- ***AI techniques implementation***—This task is a continuation of the new Phase II task for identifying additional tools for database registration. In this task the most useful tools will be identified and prototype implementations will be produced.

- ***Security study***—In this task we will develop an overall approach for security enforcement in ANSWER and make recommendations for implementation and exploration of key concepts.

- ***Demonstration and training***.

- ***Distributed query processing***—In this task we will identify existing commercially available distributed query processing capabilities and design the integration of ANSWER tools with the distributed query processing capability.

1.3.4 Phase IV

Phase IV represents the remaining six months of the original Phase III of the contract. The major task in Phase IV is the implementation and integration of the ANSWER tools with existing Army distributed query processing facilities. The scope of these tasks will be discussed with the Army in 1990. The tasks include:

- ***Distributed query processing***—This task will implement the integration of ANSWER tools with a distributed query processing facility.

- ***Demonstration and training***—Install the ANSWER system on hardware at an Army installation. Conduct training sessions in the installation, use and maintenance of the ANSWER system.

1.4 Report Outline

The rest of this report is organized as follows. Section 2 discusses the system design for ANSWER, including the information architecture, the representation model and the software architecture for the ANSWER system.

Section 3 discusses the encyclopedia manager, one of the two major software deliverables for Phase I. The manager consists of a set of tools to create, store and manipulate schema objects and ADD elements. The encyclopedia manager includes a schema integrator, IRDS running on ORACLE, input and output schema processors that provide communication between IRDS and the schema integrator, and tools to aid in the creation and modification of ADD associations with schema objects.

Section 4 discusses the other major Phase I software deliverable, a graphical Browser for the ANSWER information architecture. This section also discusses the implementation environment for the Browser.

Section 5 discusses the results of a study we performed on possible AI enhancements to the ANSWER tool set. It discusses some different approaches to schema integration and some possible enhancements to the Browser based on AI techniques.

Section 6 summarizes the contents of this report and discusses the future plans for the ANSWER program.

Section 2
System Design

The overall ANSWER systems includes both an information architecture and a tool set architecture. This section describes both of these and discusses how the Phase I deliverables form a subset of the envisioned architecture.

2.1 Information Architecture

The ANSWER information architecture describes the major types of data descriptions required by ANSWER to support Army personnel in locating data managed by ANSWER. The Phase I information architecture includes the levels of description, shown in Figure 2-1.

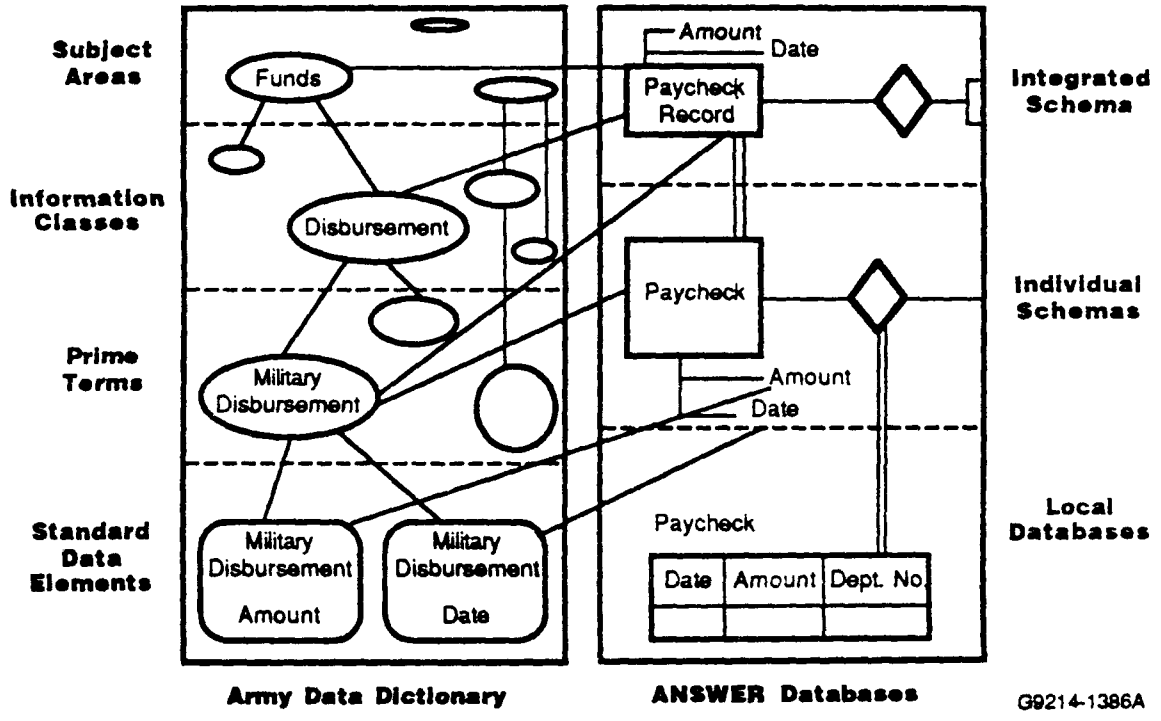


Figure 2-1. Phase I Information Architecture

The first three levels of information reflect the categorization of data as used in the Army Data Architecture and discussed in Army regulation AR 25-9. That document includes mapping relationships between subject areas, information classes and prime words. Those mapping relationships have been strictly adhered to for the Phase I ANSWER deliverables.

Subject areas describe general groupings of information classes. They describe a logical area of information used by the Army.

Information classes describe some subset of the information in a subject area. An information class corresponds to information produced from some process in the HQDA information model.

Prime terms and *data elements* are those portions of the Army Data Architecture that can be directly associated with a schema object. Prime terms are associated with individual entities, categories or relationships at the integrated and/or single schema levels. Data elements are associated with attributes of schema objects.

This information architecture will support data base registration of new databases into the ANSWER system and integration of new schemas with existing schemas. The schema integration tool, discussed later in this section, will produce an integrated schema using two or more individual schemas as input. Users can brows the five-level architecture to determine exactly where information to satisfy their queries is to be found. The ANSWER Browser, also discussed later in this section, supports browsing at all five levels of the ANSWER information architecture. It presents a graphical display of the information contained at each level and allows the user to scroll the screen or randomly access individual nodes by name or by mapping relationships to nodes at other levels.

The final ANSWER information architecture should include an enterprise-level model parallel to the subject-area, information-class, and prime-term levels. The enterprise-level model, called a domain model, models information at a level of abstraction that corresponds to the end user point of view. This model will allow the user to formulate a question without regard to location or specific logical representation of data within the ANSWER system. The complexity of such a model is directly related to the ease with which a user may formulate a query and the extent to which the user must understand any of the details of the logical and/or physical representation of the data.

The schemas are all represented in the entity-category-relationship (ECR) data model [ELMA85]. The ECR data model is an extension of the entity-relationship (ER) model originally proposed by [CHEN76]. The ECR model views the world as consisting of entities and relationships among entities. Entities and relationships have attributes that describe them. Individual entity instances are grouped together into entity types. A category is is a subset of entities from an entity type. Categories allow the representation of generalization hierarchies. In Figure 2-2, ENLISTED and OFFICERS are categories of the MILITARY-PERSONNEL entity type. Categories share attributes with the entity type on which they are defined. Thus both ENLISTED and OFFICER have attributes RANK, NAME and SOCIAL-SECURITY-NUMBER. In addition, each category may have additional attributes. The category ENLISTED also has the attribute DATE-ENLISTED, and OFFICER has the DATE-COMMISSIONED attribute. The category modeling notion is very similar to the notion of subtype in an object-oriented data model or frame-oriented model.

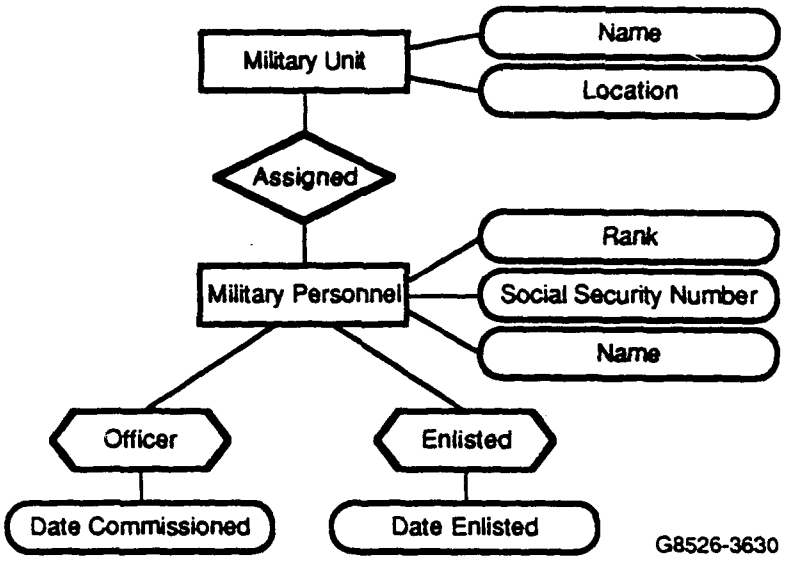


Figure 2-2. Example Entity Type: MILITARY PERSONNEL

Schemas expressed in terms of the relational data model, network or the hierarchical data model may be expressed in terms of the ECR model. This is an important point since the Army's databases will be expressed in terms of one of these models and not in terms of the ER or ECR models.

Relational schemas are relatively easy to convert to ECR schemas. The most mechanical conversion is the following set of steps:

1. Each relation, R, is treated as an entity, E.

2. The attributes of E are all the attributes of R minus the attributes functioning as foreign keys.

3. Each foreign key (or set of foreign keys if composite) corresponds to a relationship arc. If the foreign key occurs in R1 and is a key in R2, then the relationship arc will be between E1 and E2. The degree on the arc is always mapped as 1-to-M, from E2 to E1.

4. Relations that are all key and that have all keys functioning as foreign keys will be represented as entities with no attributes in the ECR model.

Hierarchical and network schemas may also be translated to the ECR model through the following steps:

1. Each record type corresponds to an entity type with all the fields of that record being the attributes of the entity.

2. A set type corresponds to a 1-to-M relationship arc from E1 to E2, where E1 is the entity derived from the owner record type and E2 is the entity derived from the member record type.

3. Any repeating groups within a record must be treated as a separate entity with a single attribute corresponding to the repeating group attribute. If more than one field is involved in the repeating group, then the entity corresponding to the repeating group must have more than one attribute.

4. Any link record type, i.e., a record type introduced solely for the purpose of capturing an M-to-M relationship, is treated as an entity with no attributes.

Some issues may arise in the selection of keys for record instances. In some cases, new arbitrary identifiers may need to be generated.

The mechanical translation from any of these data models to an ER or ECR model may appear to result in some nonintuitive entities, particularly in the case of all key relations, but no information is lost in the translation ([ACMP80], [KENT84], [MART84], [HULL87]). In addition, a data administrator may always modify an automatically translated schema so that it conforms more closely to the data administrator's world view.

2.2 Tool Set Architecture

The software architecture originally envisioned for ANSWER is shown in Figure 2-3.

That architecture includes the following modules:

- User interface,
- Query formulation,
- Browser,
- Encyclopedia manager,
- Distributed query processor.

The relationships between the tool set architecture and the information architecture is shown in Figure 2-4. The Browser and the encyclopedia manager access all levels of the information architecture. The query formulator will also access all levels of the architecture but will be primarily concerned with the schema level objects. The distributed query processor will be solely concerned with the schema-level objects.

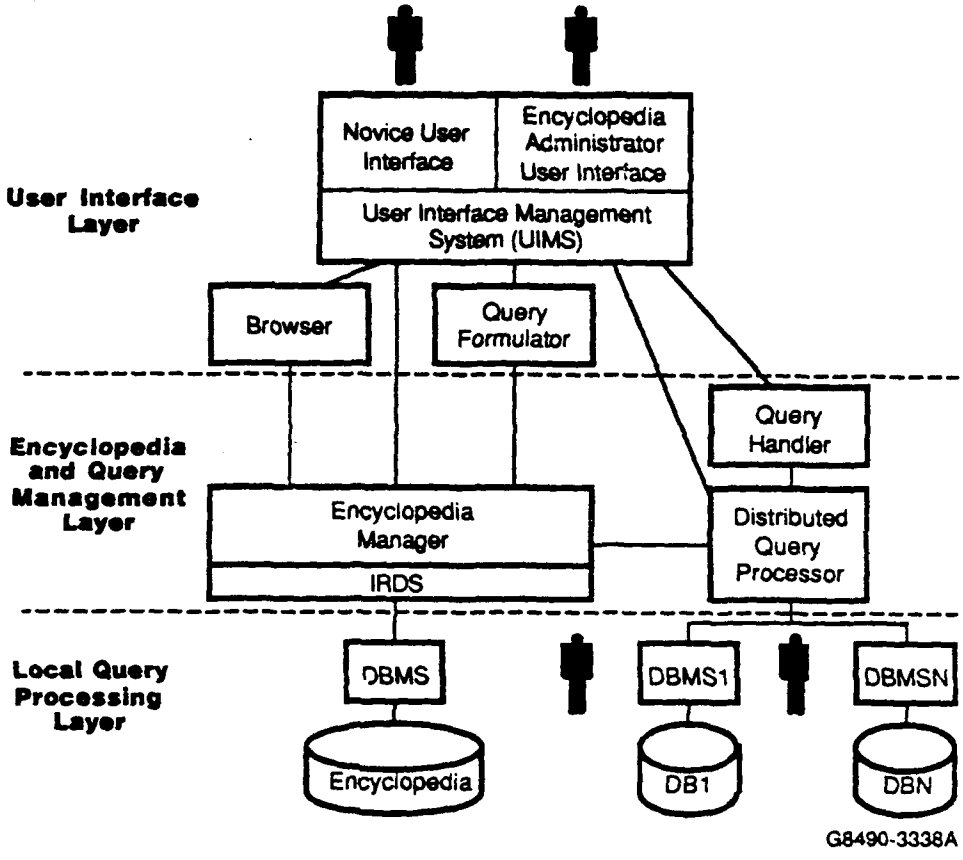


Figure 2-3. Original ANSWER Software Architecture

In Phase I, we have developed software prototypes for the Browser and the encyclopedia manager. The Browser is currently unintegrated with the encyclopedia manager, but it operates on data structures that can be interfaced to the schema and Army Data Dictionary (ADD) representations used by the encyclopedia manager. The encyclopedia manager consists of three major modules that have been completely integrated: a schema integrator, IRDS and ORACLE.

The software architecture has been designed to meet the long-term goals of the ANSWER program, providing support for interoperability, integration and synchronization of the Army's information systems. The ANSWER system will be able to meet these goals by (1) providing tools to aid in the creation of integrated database schemas, (2) having appropriate associations with Army data dictionary elements, and (3) providing tools to browse and query these schemas to identify appropriate information for end user queries.

Some of the Army's short-term goals dovetail well with the ANSWER deliverables. One of the short-term goals of the Army Data Management Division is to develop an Army Data Dictionary. The ANSWER encyclopedia manager and Browser can be

used in support of this goal. The encyclopedia manager provides facilities for the data administrator to enter and modify associations between Army Data Dictionary elements and schema elements. The encyclopedia manager also provides facilities to store and retrieve newly created standard data elements through the IRDS facilities. The utility of these tools for the Data Management Division's goals will be evaluated as part of Phase II.

Section 3
Encyclopedia Manager

3.1 Introduction

This section discusses the tools associated with management of the ANSWER information architecture. This set of tools is referred to as the encyclopedia manager. It includes:

- The Information Resource Directory System (IRDS) running on the ORACLE relational database system,
- The schema integrator tool,
- Input and output schema processors.

Figure 3-1 displays a high-level view of the functional architecture of the encyclopedia manager.

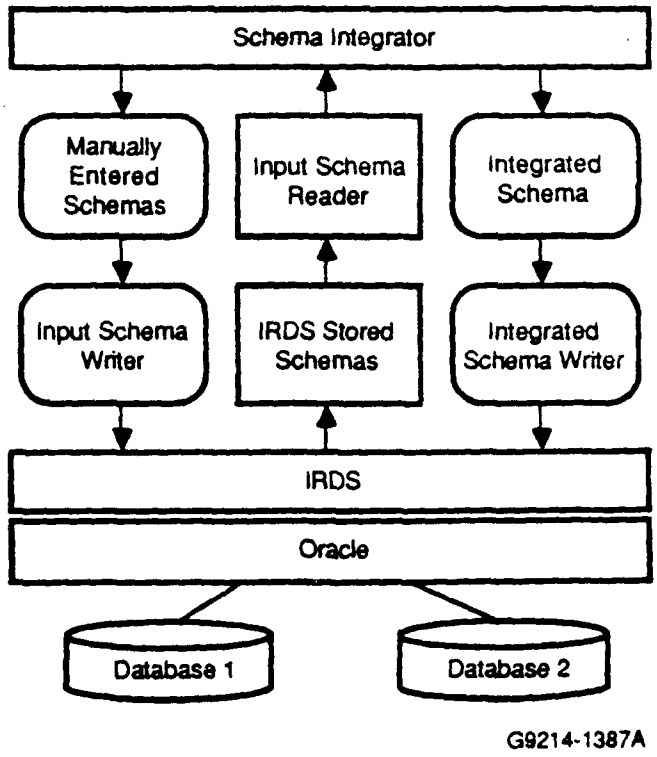


Figure 3-1. High-Level View of the Functional Architecture of an Encyclopedia Manager

These tools support the creation and management of the ANSWER information architecture. The Army Data Dictionary, represented in IRDS, contains descriptions of all subject areas, information classes and standard data elements supported by ANSWER. Individual and merged schemas are also stored in the ORACLE database and can be accessed either through IRDS or directly through ORACLE. The schema integrator reads schemas from IRDS for integration, produces an integrated result and writes it back to IRDS. The interface between the schema integrator and IRDS is managed by the input and output schema processors.

The encyclopedia manager tool set supports the following functions:

- Create, modify, delete, find, and list Army Data Dictionary elements and schema objects.
- Support the definition deletion and modification of both syntactic and semantic mappings between ANSWER information architecture levels.

New ADD elements can be entered by interacting directly with IRDS. New schema objects can be entered through the schema integrator tool. Mapping information can be directly entered into IRDS or produced and stored in IRDS automatically as a result of schema integration.

Section 3 discusses further details of the architecture and implementation of each of these tools.

3.2 IRDS

The Information Resource Dictionary System (IRDS), developed by the National Institute for Science and Technology [NBSIR 88-3700], is used to store the contents of the Army Data Dictionary (ADD) and the individual and integrated schemas represented in ANSWER. IRDS was developed to provide facilities for recording, storing, and processing descriptions of an organization's significant data and data processing resources. It is the result of an effort to develop standards for dictionary software, initiated by both the American National Standards Institute (ANSI) and the National Institute for Science and Technology (NIST).

Execution of IRDS requires the ORACLE database management system. The information in IRDS is stored in relational schema in ORACLE. All IRDS commands are translated to SQL commands executed against ORACLE relational schemas. A partial example is shown in Figure 3-2.

The relation shows an example of how IRDS information is represented in ORACLE.

TABLE NAME: ENTY_ATT	
(ENTITY_TYPE	CHAR (64)
ENTITY_NAME	CHAR (32)
VAR_NAME	CHAR (8)
REV_NUM	INTEGER
DESCRIPTIVE_NAME	CHAR (64)
ADDED_BY	CHAR (32)
ALLOWABLE_VALUE	CHAR (32)
CLASSIFICATION	CHAR (32)
CODE_LIST_LOCATION	CHAR (32)
COMMENTS	CHAR (240)
DATA_CLASS	CHAR (32)
DATA_TYPE	CHAR (16),
DESCRIPTION	CHAR (5000)
DICT_PARTITION_NAME	CHAR (32)
DOCUMENT_CATEGORY	CHAR (32)
EXTERNAL SECURITY	CHAR (32)
INTERNAL_FORMAT	CHAR (32)
IRD_SCHEMA_PHASE_NAME	CHAR (5)
JUSTIFICATION	CHAR (32)
LAST_MODIFIED_BY	CHAR (32)
LENGTH	INTEGER
LOCATION	CHAR (32)
MOD_COUNT	INTEGER
INTEGER_OF_RECORDS	INTEGER
NUM_LINES_CODE	INTEGER
PRECISION	INTEGER (2)
RECORD_CATEGORY	CHAR (32)
SCALE	INTEGER (2)
SYSTEM_CATEGORY	CHAR (32)
USAGE	CHAR (32))

G9082-1430

Figure 3-2. A Partial Example of the ORACLE Scheme for IRDS Information

IRDS uses the entity relationship (ER) data model. The Army Data Dictionary is represented in IRDS using that data model. A relational schema for the ADD exists and is in use at the Data Management Division. The ADD model represented in IRDS for ANSWER is based on that relational schema. The ADD ER representation in IRDS is shown in Figure 3-3.

The ADD is used to manage standard data elements. It also includes information about subject areas, information classes, prime words and reference elements. Any ADD information may be retrieved from IRDS using IRDS ad hoc query facilities.

The individual and integrated schemas are represented in IRDS as entity-category-relationship models [ELMA85].

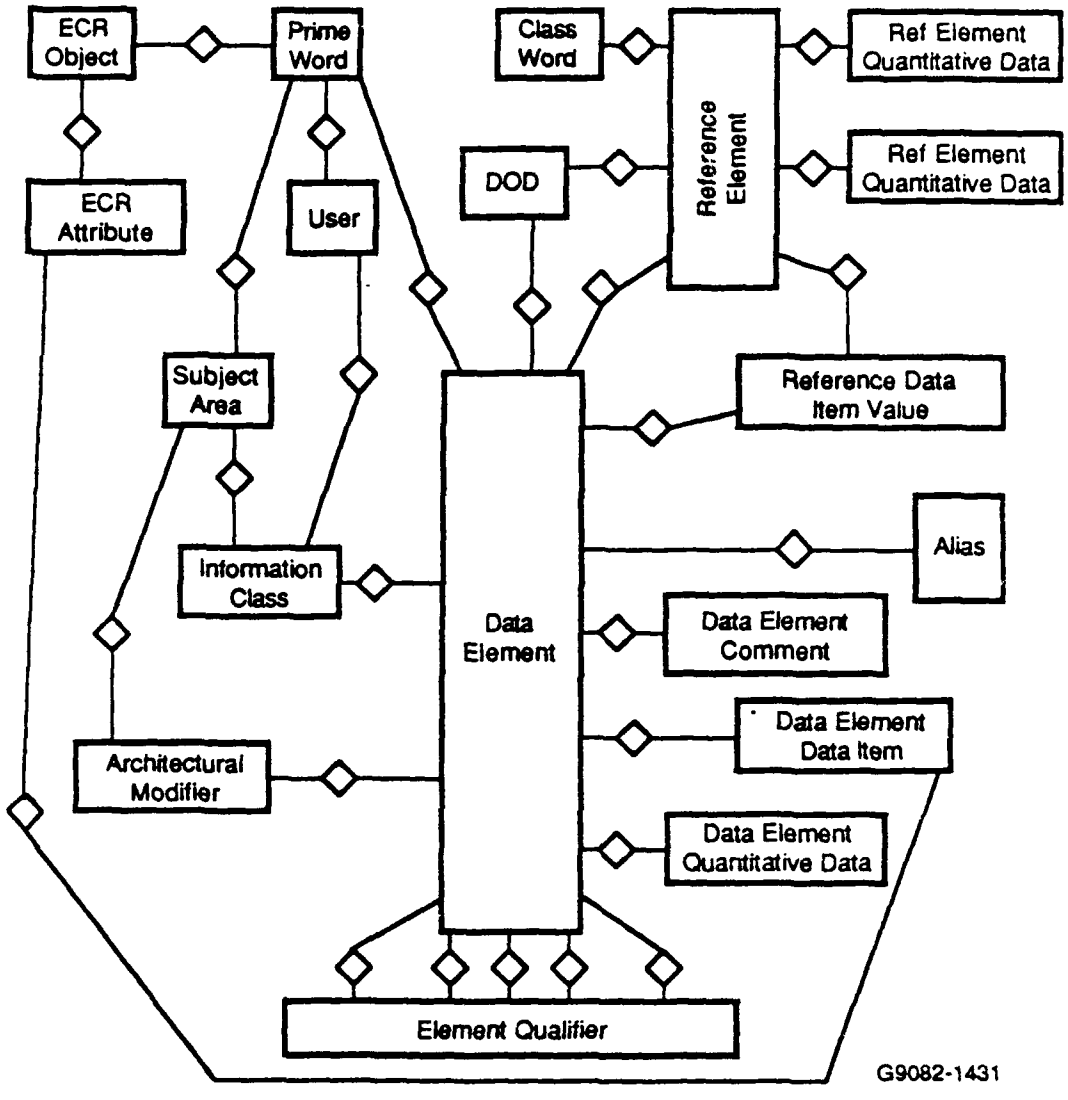


Figure 3-3. Add ER Representation in IRDS

The category concept allows subtype information to be explicitly represented in the schemas. Subtyping is a critical formal representation technique for schema integration. Many schemas contain objects whose extensions partially overlap with other schemas' objects or which are wholly contained in or contain the extension of other schemas' objects. In such cases, the category concept allows that relationship to be formally captured. For example, if a database containing an Installation object is integrated with a database containing a Military_Base object, the Military_Base object would be most properly represented as a subtype, that is a category, of the Installation object.

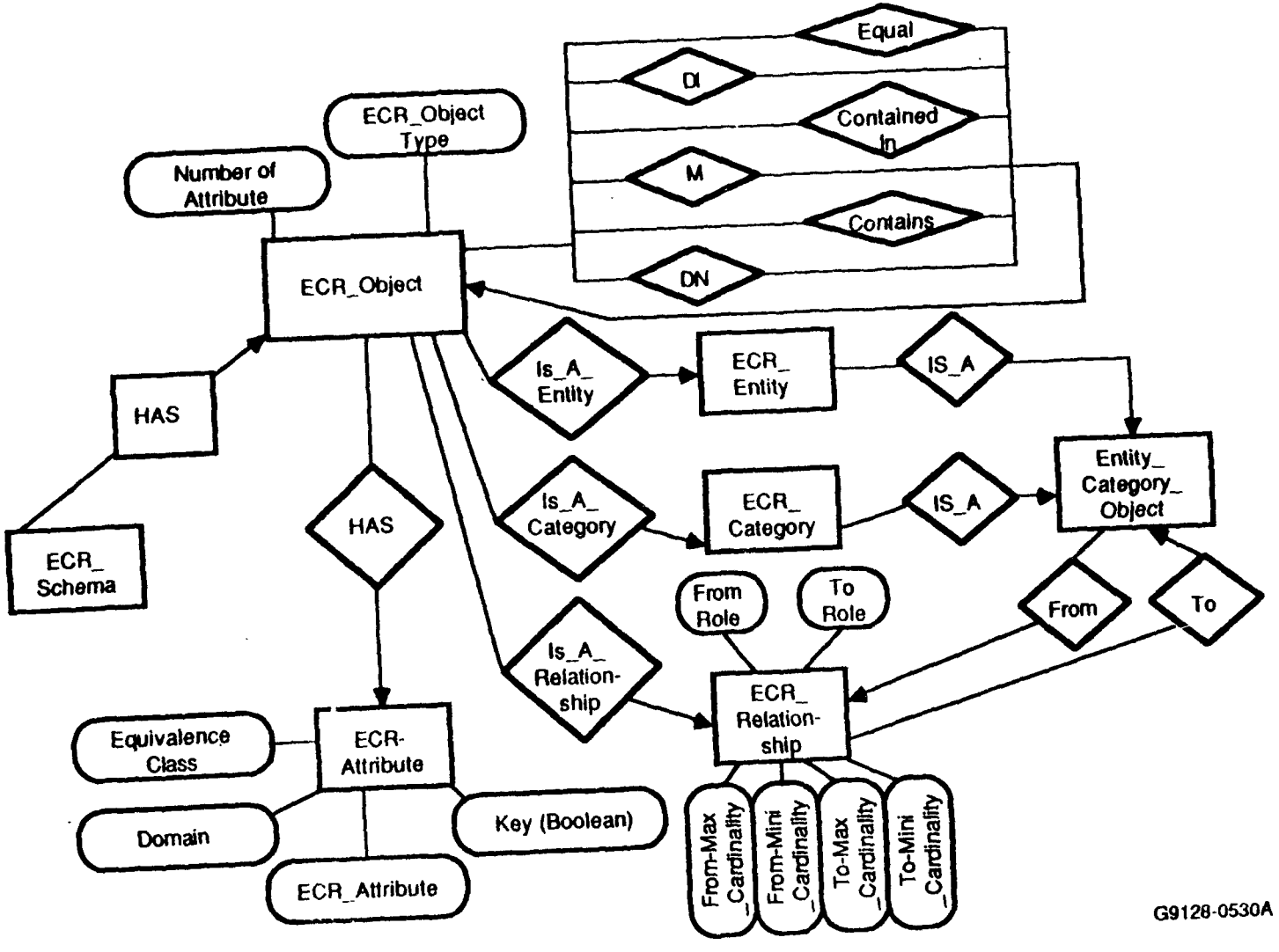


Figure 3-4. ECR Model as Represented in IRDS in Terms of the ER Model

The ECR model is represented in IRDS in terms of the ER model as shown in Figure 3-4. Schemas are then represented as instances of the ECR entity and relationship types in IRDS.

The ECR model is represented as a collection of entities. The ECR entity is represented as an entity called ECR-entity. Similarly relationships and categories are represented as entities called ECR-relationship and ECR-category. The ECR-relationship entity relates two entities or categories as indicated by the From and To relationships to an entity called Entity-category-object. Relationships also have additional information associated with them represented here as the attributes of the entity ECR-relationship.

For purposes of schema integration, ECR entities, categories and relationships are all treated as ECR objects. The entity ECR-object is related to an entity ECR-attributes, instances of which correspond to the schema attributes for the schemas to be integrated.

3.3 Schema Input/Output Processors

The schema processors are required to read and write schemas from the schema integrator local file structures to the IRDS storage structures. As shown in Figure 3-5, the schema processors read information directly from ORACLE via relational queries against the ORACLE schema. The information is structured into four local files for the schema integrator describing the schema's entities, categories, attributes and relations. The schema output processors write information back out to ORACLE through the IRDS interface. Information may be written out at two different times:

- After entering new schema information into the system through the schema integrator interface.
- After completing the creation of a new integrated schema through execution of the schema integrator.

There are two separate schema output processors because the schema integrator maintains two different local file structures to represent schema information. Before the actual integration process, schemas are represented locally in the schema integrator in a different set of files than are used after the integration process is complete. Depending on the point at which the schema is to be written out to IRDS, it is represented using one of these two file structures.

3.4 Schema Integrator

The schema integrator is the primary tool for creating the integrated schema layer of the ANSWER information architecture. The integrated schema will allow users to query the ANSWER system for information without regard to the logical or physical

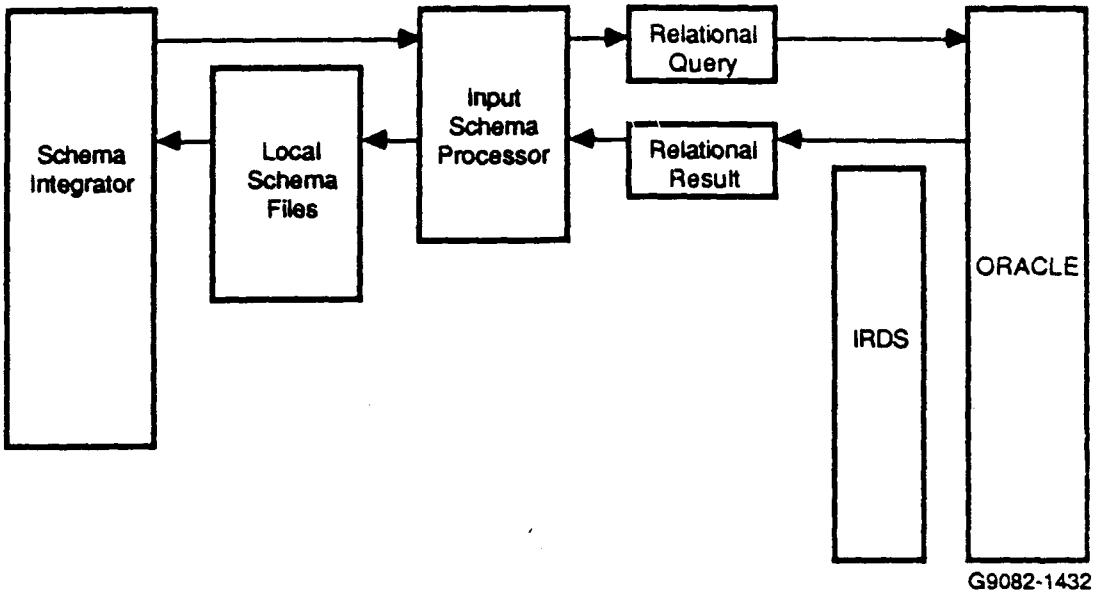


Figure 3-5. Schema Processor

location of the data. It is an essential component of the ANSWER information architecture. Without an integrated schema users would need to be cognizant of the details of all schemas to which they required access as well as the relationships between each individual schema.

The schema integrator takes individual schemas and assertions about schema relationships as input and produces an integrated schema as output. The resulting schema represents inferred equality and subtyping relationships between schema objects in a single structure. It also includes all of the information contained in the original input schemas, in some cases augmented with additional abstractions to indicate schema relationships. The most commonly introduced abstractions are:

- Renamed objects,
- Newly introduced objects,
- Demotion of entity to category with a new subtype relationship.

Objects in one schema may be renamed in an integrated schema if there is an equivalent object. For example, if schema 1 has an entity *Base*, and schema 2 has an entity *Military_Base*, each having exactly the same attributes, the resulting integrated schema will have only one of the two entities represented. If Military_Base has one additional attribute, not included with Base, and both entities are asserted to be equal, then a new entity called D_M_Base would be introduced into the integrated schema. D_M_Base would have all attributes in common between Base and Military_Base. Military_Base would still occur as a category of D_M_Base with any additional attributes.

3.4.1 Approach to Schema Integration

The approach to schema integration is based on [ELMA86]. The basic approach to integration for Phase I is to use a n-ary one-pass strategy to integrate n input schema. The advantage of this approach is that it eliminates any backtracking that may be required if a binary approach is used. The major modules of the schema integrator are shown in Figure 3-6.

The major modules are schema collection, attribute equivalence specification, object assertion specification, assertion consistency checking, pruning and partitioning and lattice merging.

Schema collection collects information on schemas to be merged and assembles it into internal file representations used by the schema integrator. The user may manually enter schema information or elect to have the information loaded directly to the schema integrator files from IRDS.

Attribute equivalence specification presents the user with attributes from each input schema and collects assertions about the equivalence class status of the attributes. The user may assert that any pair of attributes have one of the following relations:

- Equal,
- Contains,
- Disjoint and not integrable,
- Disjoint and may be integrable,
- May be integratable.

At the point where assertions about relationships between attributes are being collected, the user is also presented with standard data elements that may have been associated with the attributes. The user may modify the standard data element associated with an attribute by making it the same as the standard data element of the other attribute being examined or by selecting a new standard data element from a list presented on request. Future versions of this tool may include a tool that supplies the user with aid in creating standard data elements for an attribute.

Object assertion collection allows the user to make the same type of assertions about relationships between entities, categories, or relationships in the two schemas. Later versions of the schema integrator tool will support the association of prime terms with schema objects in the same way that standard data element association is currently supported.

Assertion consistency checking evaluates the set of user assertions looking for contradictions in the assertions. For example, it is not possible for two objects A and B to be both 'equal' and 'disjoint' at the same time.

The pruning and partitioning step of schema integration identifies distinct equivalence classes for integration. This step reduces the work of integration by

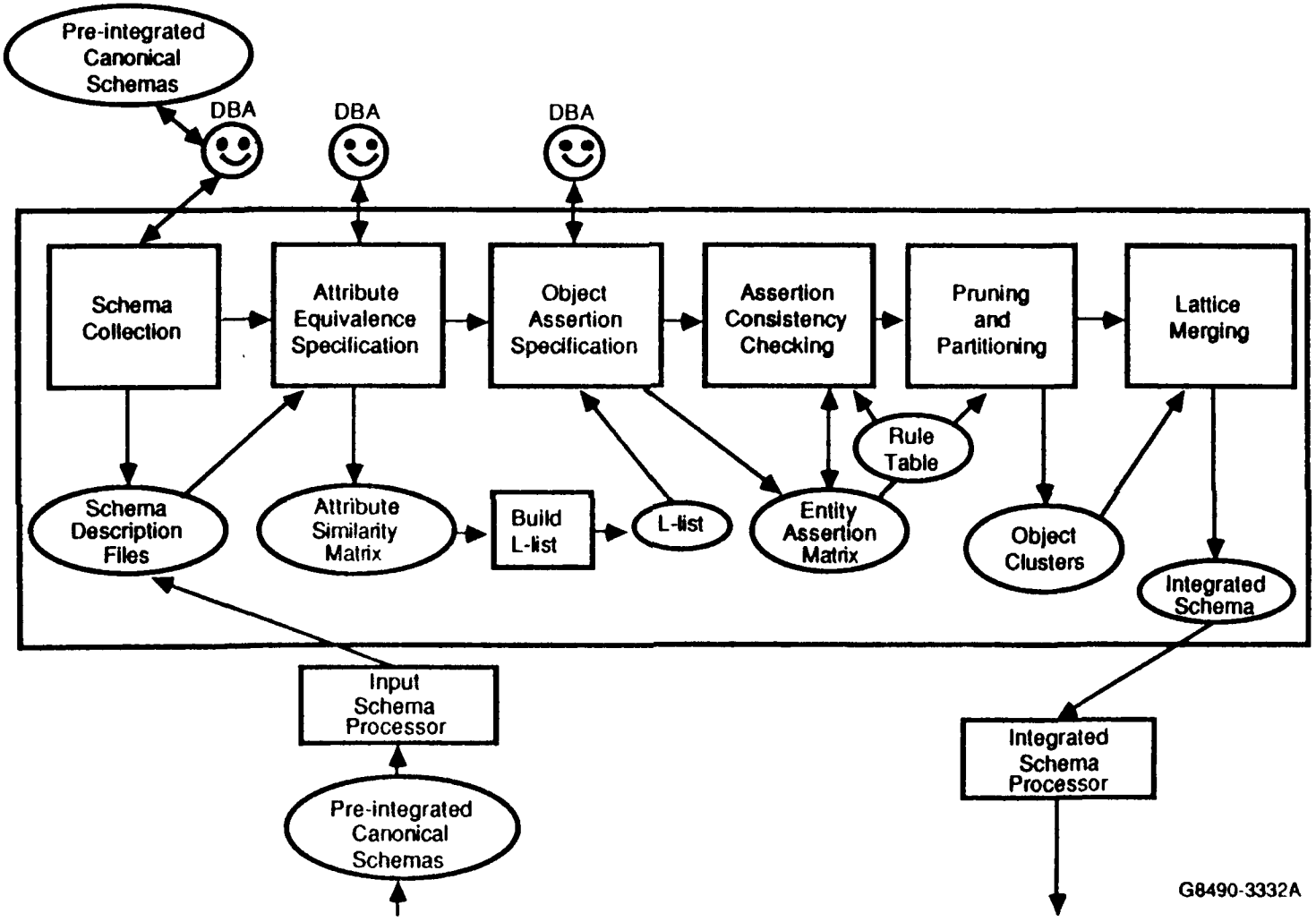


Figure 3-6. Major Schema Integrator Modules

assembling into clusters only those objects that are dependent on each other during integration. Each cluster is then integrated independently.

The lattice merging function takes each cluster and generates a lattice based on subset-superset relationships between objects. Only one of a pair of objects asserted to be equal will occur in the final lattice. The idea behind the algorithm is to generate a complete lattice that corresponds to the general case of n-ary integration. The lattice is then reduced to retain only those objects necessary to represent the integration of the N objects in the cluster.

At this point the integration process is complete. The integration results are written to local schema integrator files. The files may be written back to IRDS at the request of the user. Each of these modules is discussed in more detail in the interim technical report, December 1988.

3.4.2 Integration with the Army Data Dictionary

As a by-product of our work, we have been able to use the schema integrator as a tool to aid a data administrator in the management of information in the Army Data Dictionary. The schema integrator provides facilities for the data administrator to review the standard data elements that have been associated with attributes of the schemas being processed by the schema integrator. Figure 3-7 shows a sample display screen where the standard data elements associated with the schema attributes are displayed along with the attributes.

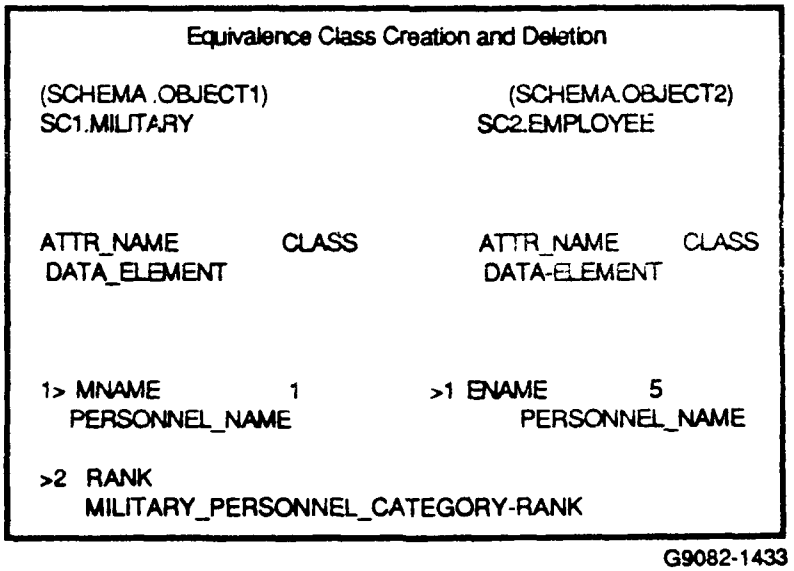


Figure 3-7. Sample Display Screen Displaying the Standard Data Elements Associated with the Schema Attributes Along with Attributes

The standard data elements may be useful to the data administrator in evaluating what attribute assertions to make during schema integration. During the attribute assertion collection phase, the data administrator must decide whether or not two attributes are equivalent. In many cases, if two attributes are equivalent, they will already have the same standard data element assigned to them. For example, schema 1 may contain an attribute called MNAME and schema 2 may have an attribute called CNAME. Both of those attributes may already be associated with a standard data element like PERSONNEL-NAME, allowing the data administrator to determine that the attributes are equivalent.

If necessary, the data administrator may assign a new standard data element to schema attributes if the current standard data element assignment is not correct. For example, schema 1 may have an attribute MNAME whose standard data element is MILITARY-STAFF-NAME, while schema 2 may have an attribute NAME whose standard data element is Personnel-name. While both Personnel-name and MILITARY-STAFF-NAME are syntactically well-formed standard data elements, PERSONNEL-NAME is semantically more appropriate and should be assigned to both attributes. The current schema integrator will allow the data administrator to change the standard data element assignment for MNAME to PERSONNEL-NAME.

Revisions of the tool will also allow the data administrator to create new standard data element associations. This is useful in cases where no association between a standard data element and a schema attribute currently exists. Currently the data administrator may view a list of standard data elements from which to choose if a new assignment is to be made. Later versions of the encyclopedia manager tool set could allow the user to invoke a tool to interactively generate the most appropriate standard data element for the attribute.

3.4.3 Example

Figures 3-8 and 3-9 show two simple schemas that can be integrated using the schema integrator.

The data administrator could load these schemas directly from IRDS or load them manually using the schema integrator schema collection module. Once the schemas are loaded into the schema integrator local files, the schema integrator will prompt the data administrator for assertions regarding the equivalence class status of all entities, categories, relationship and attributes in the two schemas.

The data administrator may wish to view the standard data elements associated with the schemas. Figure 3-10 shows a display of standard data elements.

Once all of the assertions have been collected, the schema integrator checks for the consistency of all user input and derived assertions and then performs the integration. The resulting integrated schema is shown graphically in Figure 3-11.

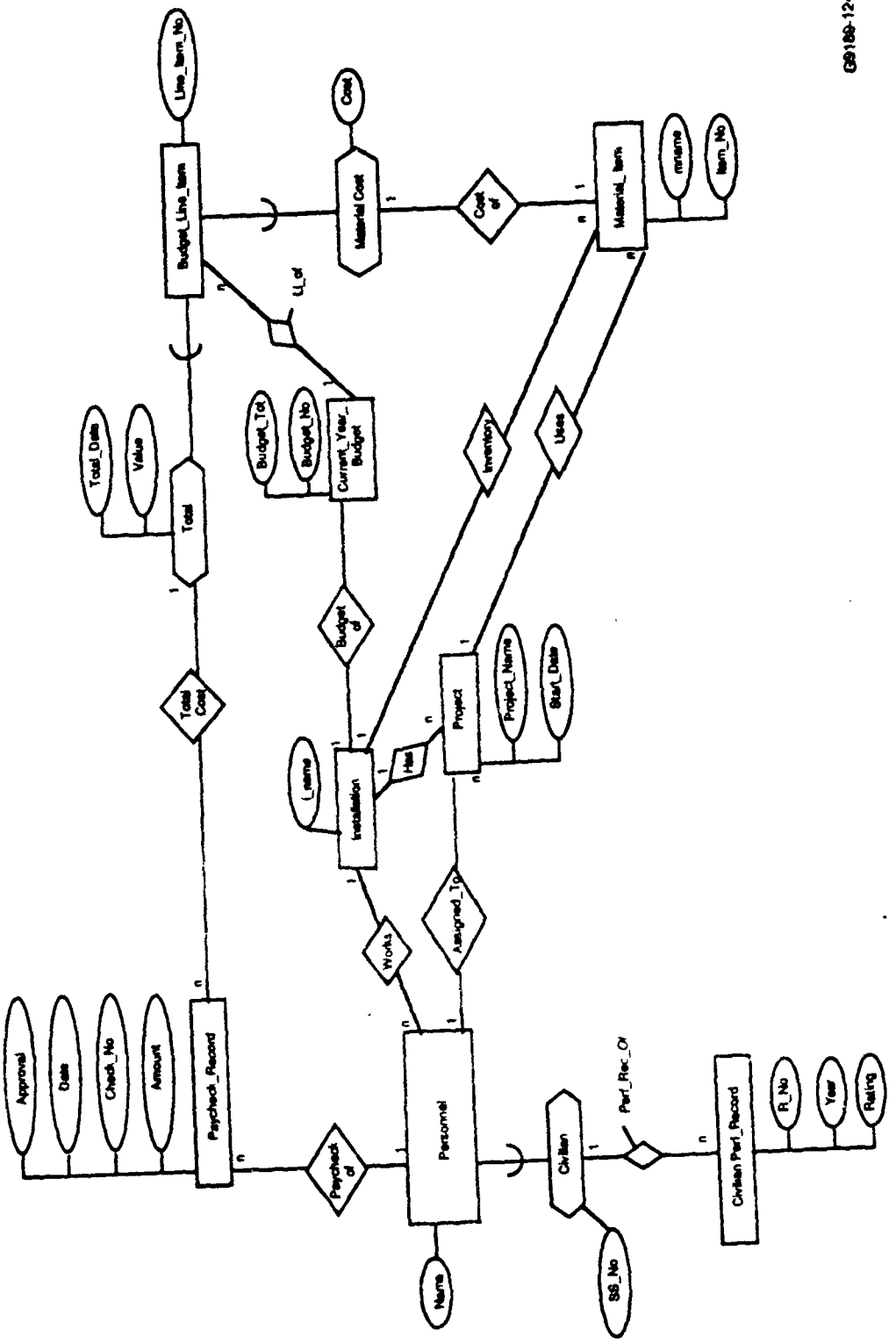
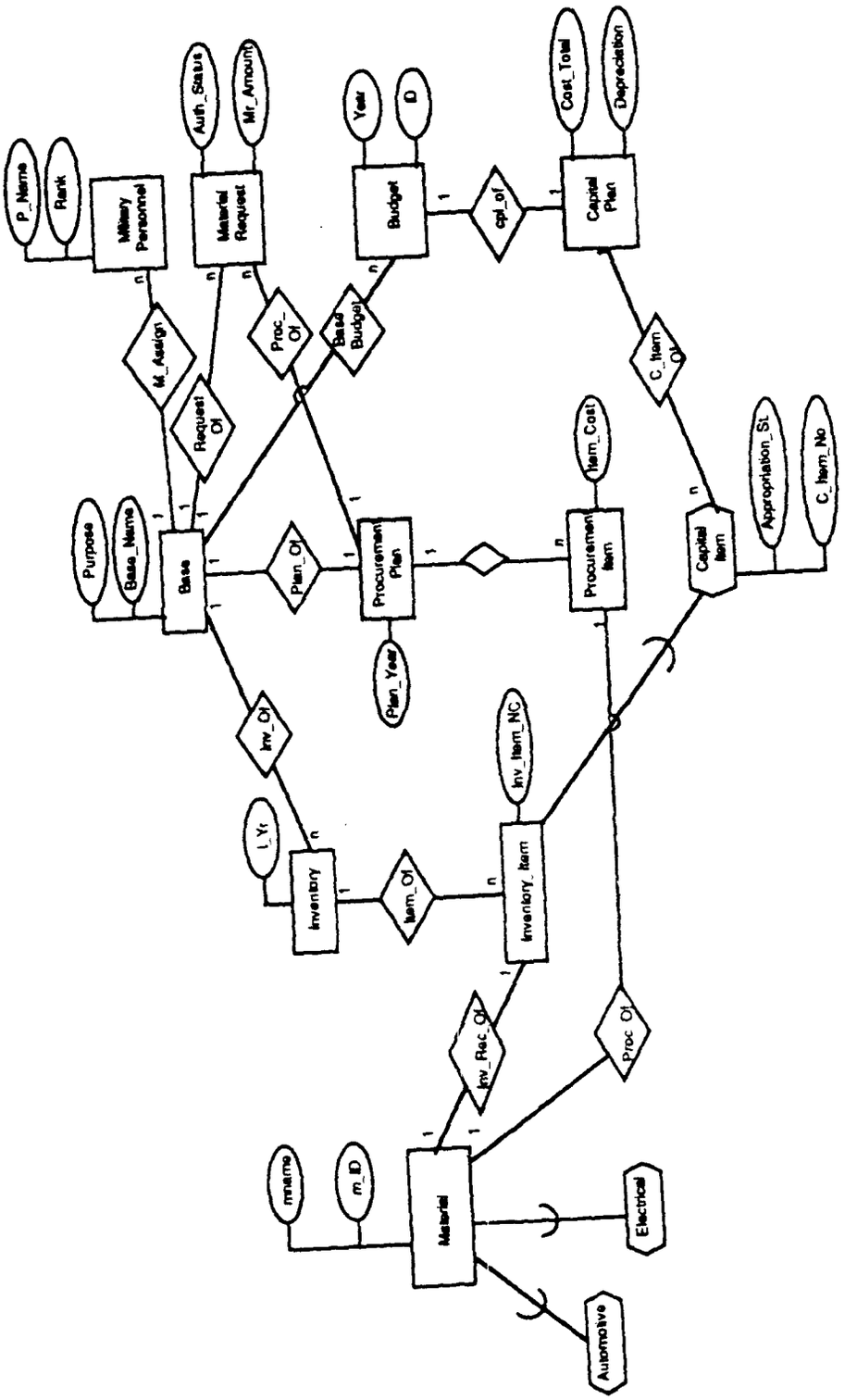


Figure 3-8. Schema 1



G89082

Figure 3-9. Schema 2

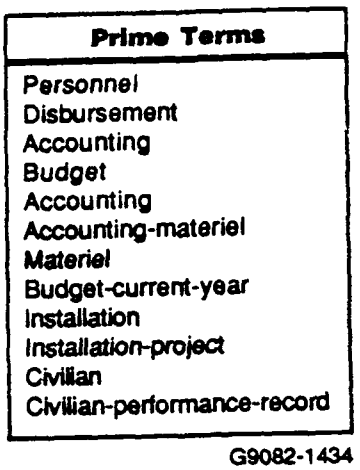


Figure 3-10. Standard Data Elements

The data administrator may then write the integrated result back out to IRDS.

3.5 Implementation

The schema integrator is implemented in C running in UNIX 3.5 on a SUN 3/60 work station. IRDS and ORACLE run in the same environment. The schema input/output processors are implemented in Pro C (an embedded SQL query language supported by ORACLE).

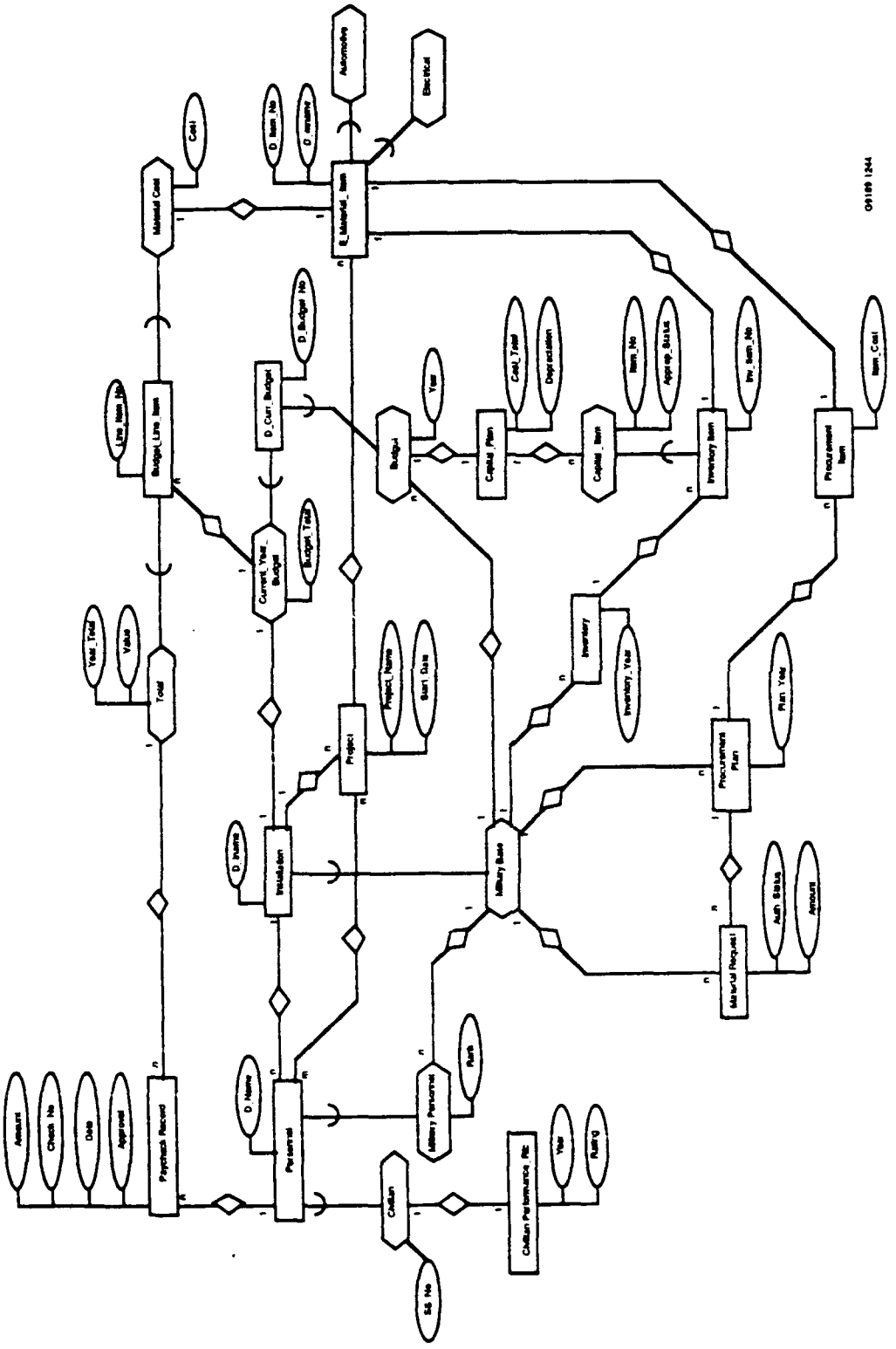


Figure 3-11. Graphical Depiction of Resulting Integrated Schema

Section 4
Browser

The ANSWER Browser is designed to support graphical browsing of the ANSWER information architecture. It displays the Army Data Dictionary (ADD) elements of Subject_Area, Information_Class, Prime_Term, and Standard_Data_Element as simple nodes in a tree structure, with lines to indicate connections between nodes. The integrated and individual schemas may be displayed as entity-category-relationship diagrams with appropriate graphical notation to indicate whether a node is an entity, attribute, category or relationship. Figure 4-1 shows an example display of some ADD elements and a schema in the Browser interface.

4.1 Motivation

The Browser allows a user to view the ANSWER information architecture from multiple points of view and to identify objects of interest for a variety of tasks including:

- Inspecting the results of schema integration,
- Identifying various elements and interrelationships in the Army information model and Army Data Dictionary,
- Investigating the relationships between standard data elements and schema objects,
- Investigating the relationships between objects in nonadjacent levels of the ANSWER information architecture.

The ability to view information from multiple viewpoints is essential to the management of large data systems. The Browser can be an aid to:

- Identifying the location of desired information,
- Understanding more detail about the semantics of a known data element,
- Understanding the established relationships between schema objects and ADD objects,
- Creating new associations between ADD objects and schema objects.

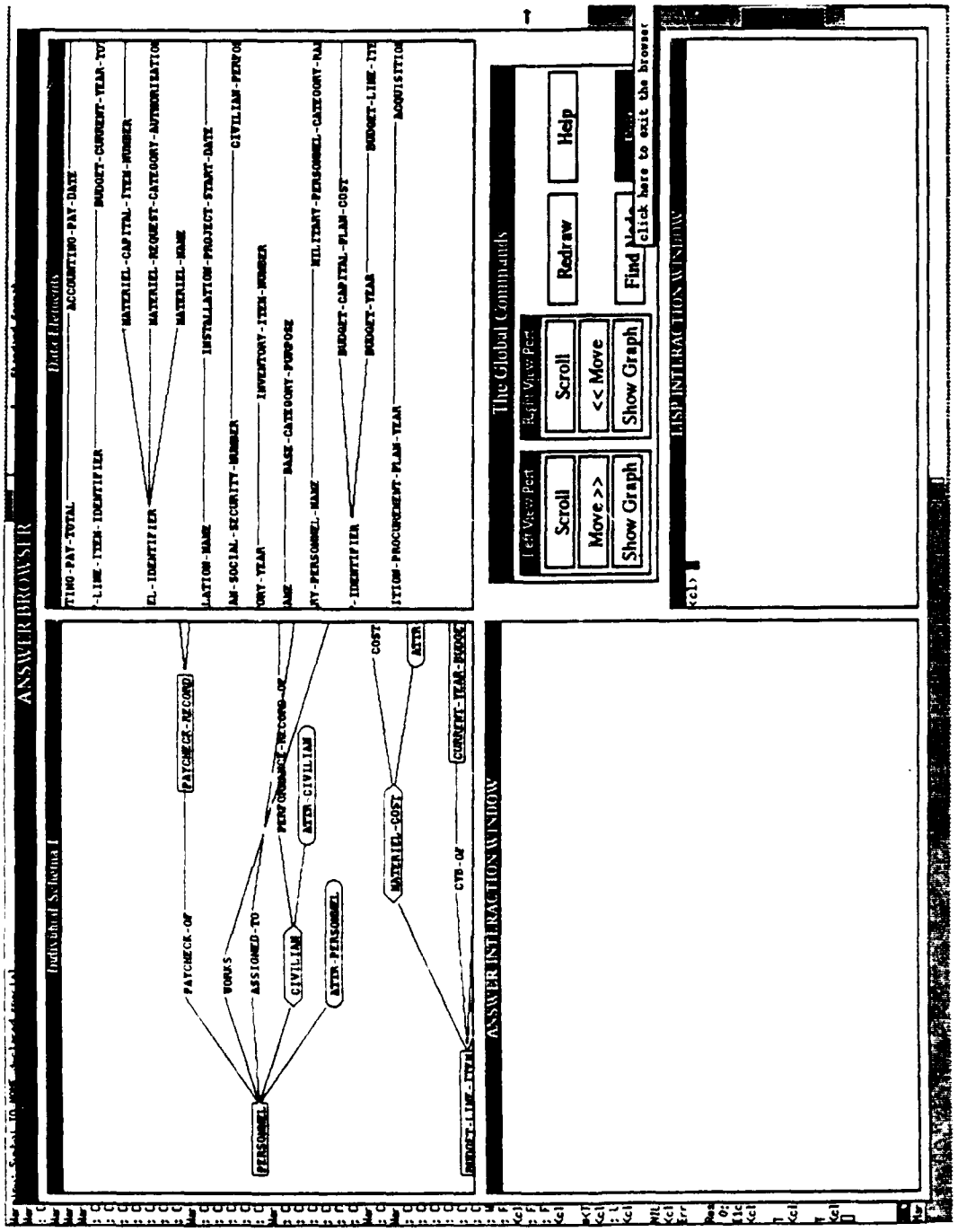


Figure 4-1. Example Display of Some ADD Elements and a Schema in the Browser Interface

The Browser support the basic notions of scroll, zoom, structuring and filtering in its graphic display functions. The user can scroll any graph displayed by the Browser in any direction by mouse manipulation. Zooming is supported via the suppression of attribute details. Attributes are only displayed upon user request and may be buried again on user request. The user may structure and filter objects by requesting certain regions of the graphs to be displayed. For example, the user may request that only the subject area terms be displayed.

The Browser also supports a topological map structure that allows the user to view the abstract structure of the graph. A example of the topological map display of a schema is shown in Figure 4-2.

This sort of display is useful for rapid reorientation of the user in the graph structure.

Users typically want to be able to index into graph structures at arbitrary levels. This aids the user in understanding the information in the context of the rest of the graph structure. Currently, individual nodes may be accessed by name or by relationship to other nodes. For example, if the user is viewing a portion of an individual schema, the user may change levels of the display and view nodes at the integrated schema, prime term, information class or subject area levels. The displayed nodes at the chosen level will be nodes related to the current individual schema node. For example, if the current node is an individual schema entity, e.g., Paycheck_Record, the nodes displayed at the prime-term level might be Disbursement_Record. The user may move from nodes at any level to related nodes at any other level. This type of display is necessary for understanding the relationship between schema objects and elements of the Army Information Model and the Army Data Dictionary. Future versions of the Browser will support node indexing via description as well as name and relationship to other nodes. The descriptions may be supported by a restricted formal language or some version of an English sublanguage sufficient to support browsing in a specified domain.

4.2 Approach

The Browser is designed to display two graph structures at a time. Each node in the graph structure has a set of commands associated with it. The user may use the mouse to click on a node to see a set of commands available at that node. Different commands are available depending on the node type.

The nodes of a schema are depicted differently depending on the node type. Entities are displayed enclosed in rectangles. Attributes are displayed in boxes with rounded ends and categories are displayed in boxes with pointed ends. Relationships are represented simply as labelled arcs. All ADD elements, (i.e., subject areas, information classes, prime terms and standard data elements) are displayed as text string nodes with unlabeled arcs between the nodes.

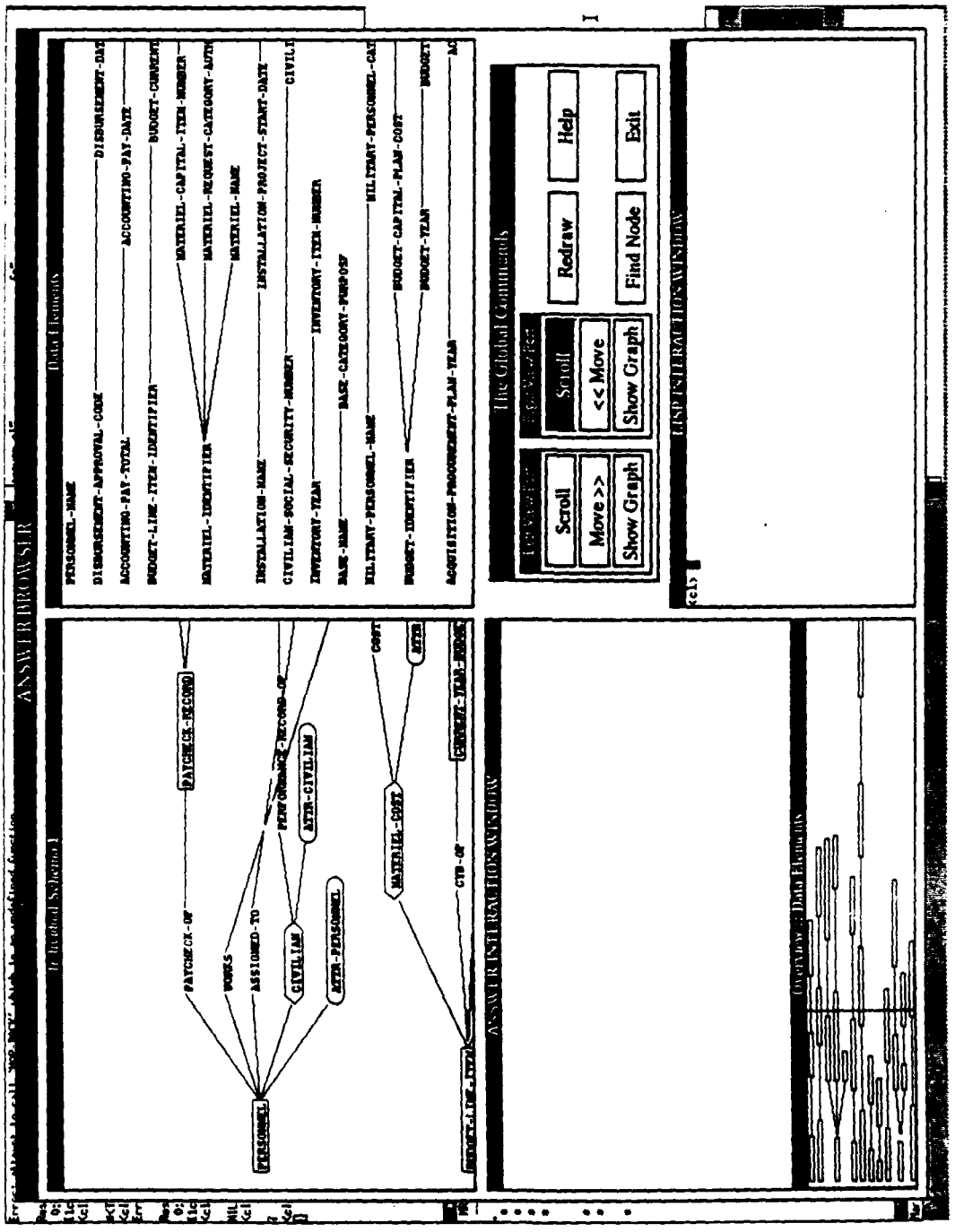


Figure 4-2. Example Topological Map Display of a Schema

The Browser is structured as a set of windows with different functions. Figure 4-3 shows a sample Browser display. The largest windows are the graph display windows. Any of the schemas or subparts of the ADD may be displayed in those windows. The smaller windows on the left are for the Browser functions that are independent of individual nodes and commands to control the graph display windows.

The set of functions supported by the current version of the Browser include:

- List attributes,
- Center this node,
- Create standard data element association,
- Show information,
- Change level,
- Find node,
- Scroll graph.

The List Attributes command presents a list of all attributes associated with an entity, category or relationship. The attributes are not displayed directly on the screen at all times. Frequently an entity, category or relationship may have too many attributes for this to be a practical option. All attribute displays are suppressed unless specifically requested by the user.

The command, Center this Node, simply redisplays the current graph with the current node at the center of the display.

Create Standard Data Element Association provides a facility to create new mappings between attributes in an individual or integrated schema and nodes of the data element structure accessible by the Browser. The association may be done graphically by using the mouse to click on nodes that are to be associated with each other.

Show Information currently displays only a definition for any node being browsed. Other information may be added as necessary. For example, attributes may also use this option to indicate domain and data type information.

The Change Level command allows the user to jump from a node at a portion of the information architecture to a related node in a some other part of the information architecture. For example, if a user wanted to display all individual schema objects associated with a a particular prime term, the command would supply all options of nodes to display at the single schema level associated with the current prime-term node. The user would be able to select of those nodes for further browsing. It is possible to invoke the Change Level command to move from a node at any level to a node at any other level in the information architecture.

The command Find Node currently supports random access to any node in any level of the information architecture. As noted earlier, later versions of the Browser will support more powerful node indexing structures.

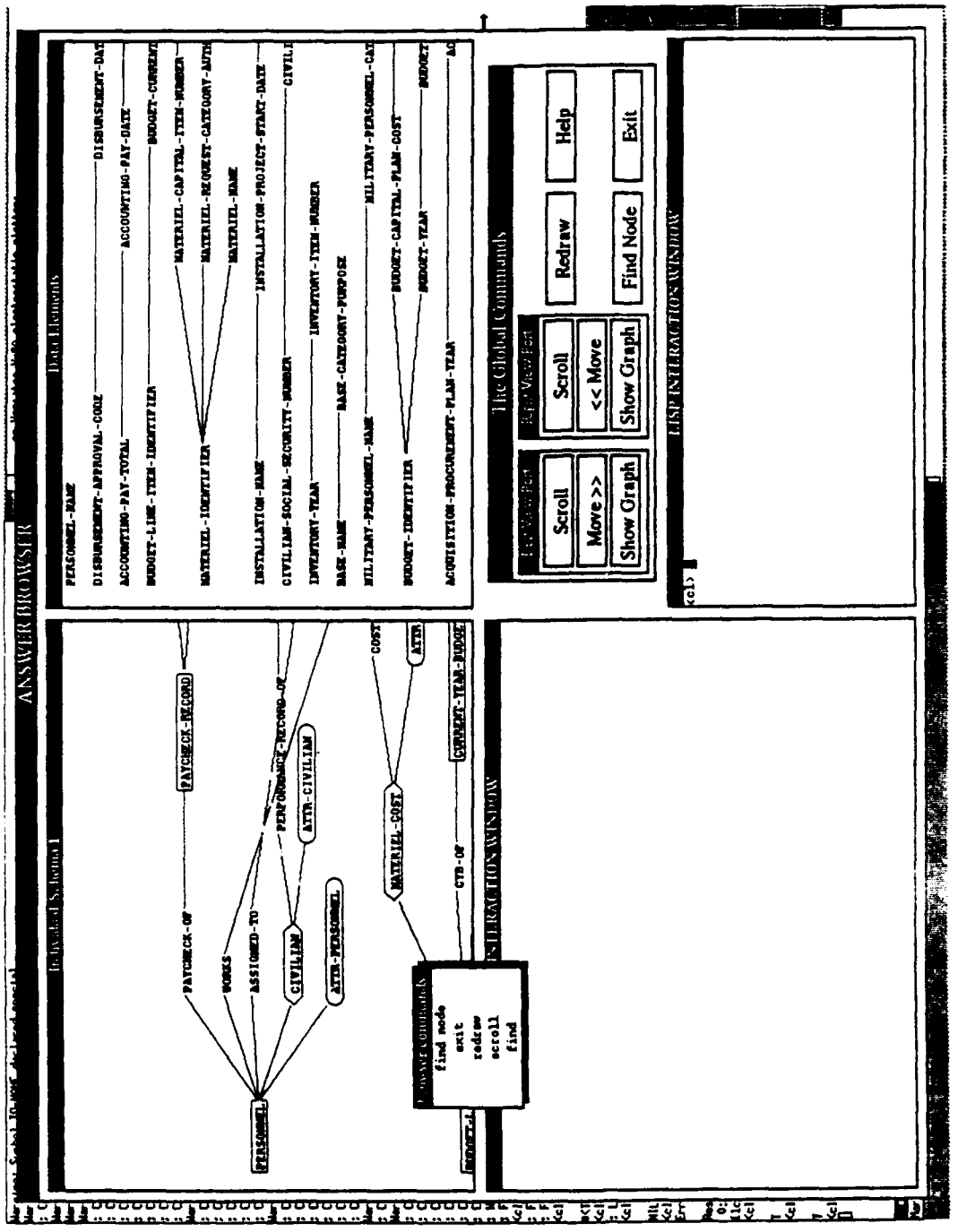


Figure 4-3. Sample Browser Display

Scroll Graph simply supports scrolling of the graph display. There is also an abstracted graph display that presents the graph as set of unlabeled nodes. This display allows the user to position themselves within the graph structure via topological cues.

4.3 Example

Figures 4-4 to 4-6 show a series of Browser windows requesting a schema to be displayed, exposing the attributes of a node in the schema, and displaying the associated standard data elements for one of the exposed attributes.

The Browser also allows the user to obtain information about the node beyond the name of the node and the graphical relationships between that node and other nodes. Figure 4-7 shows an example of a definition being displayed for a subject area.

4.4 Implementation

The ANSWER Browser is currently written in Common LISP. It uses a version of ISI-Grapher, X-Windows and Common Windows. It runs on the Sun 3/60 workstation.

ISI Grapher is used to provide the basic graph manipulation and layout functions. ISI Grapher is public domain software from the University of Southern California Information Sciences Institute. It is written in Common LISP and is running on Franz Common LISP for this demonstration system. It adheres to the Common LISP standard as described in [STEE84]. Franz Common LISP uses Common Windows, a windowing system for LISP implemented on top of X-windows.

The Browser has an event-driven control structure. It waits for an event to occur (typically a mouse click) and performs whatever action is pending. This allows the user to interact with the Browser in an unstructured manner.

The Browser node structure is defined to support the association of additional information. Currently, the node structure allows for definitions to be stored with the node. This means that the user may look at the graphical structure associated with ADD elements for example and request definitions of individual nodes to be displayed on demand. Attribute nodes for schemas also store information about an attribute's characteristics, in particular whether the attribute is a key, its domain, and data type.

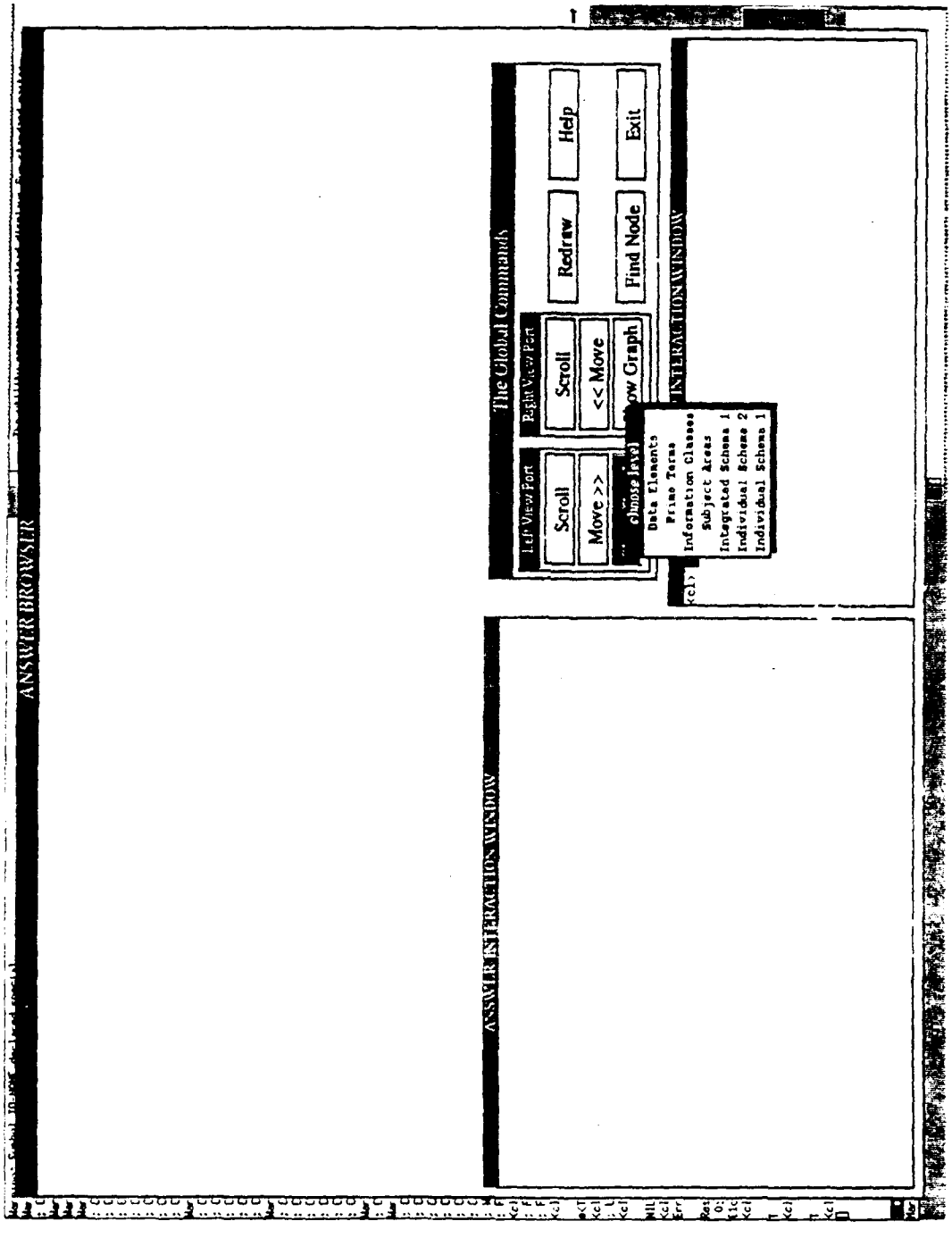


Figure 4-4. Browser Window Requesting a Schema to be Displayed

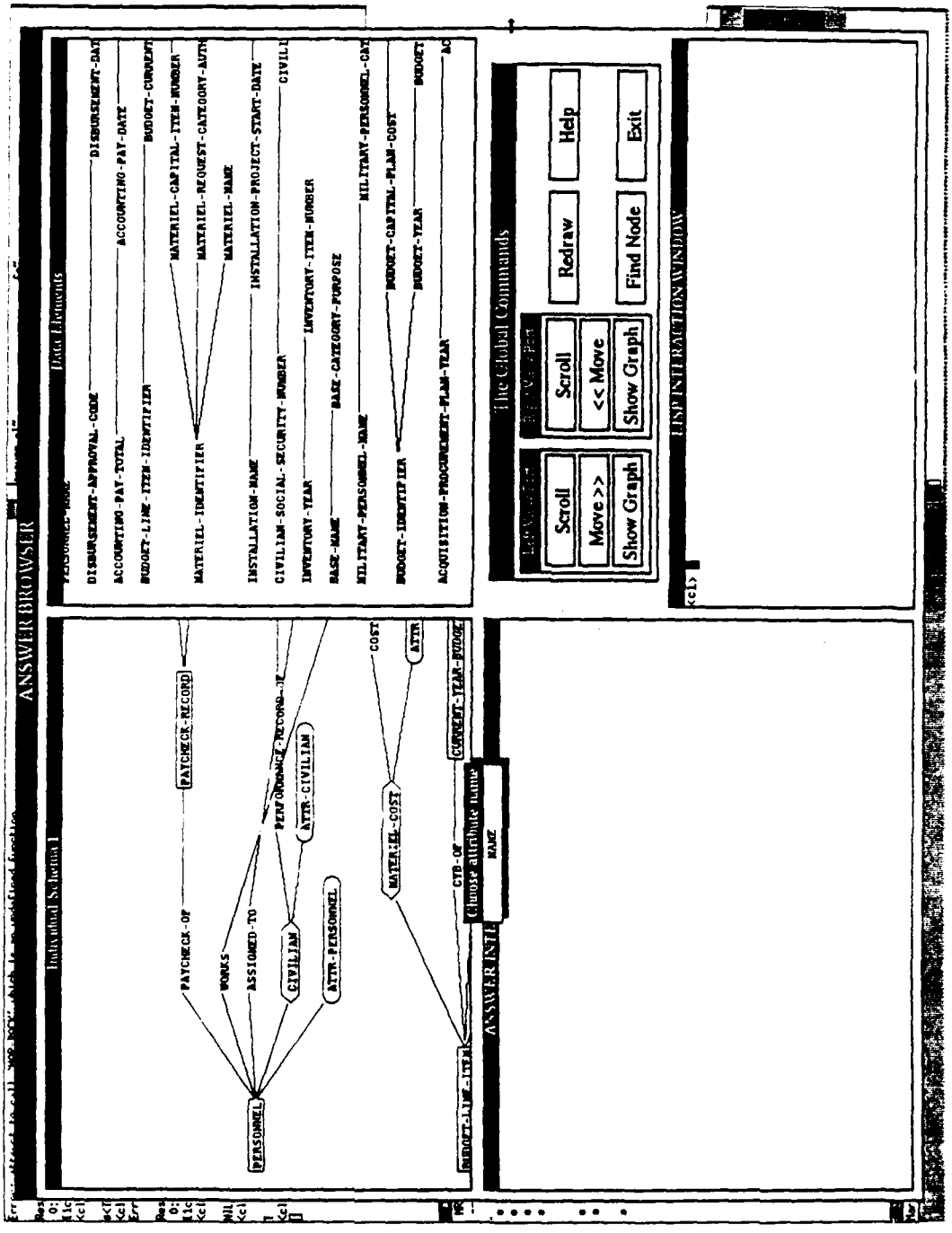


Figure 4-6. Browser Window Requesting a Schema Exposing Schema Node Attributes

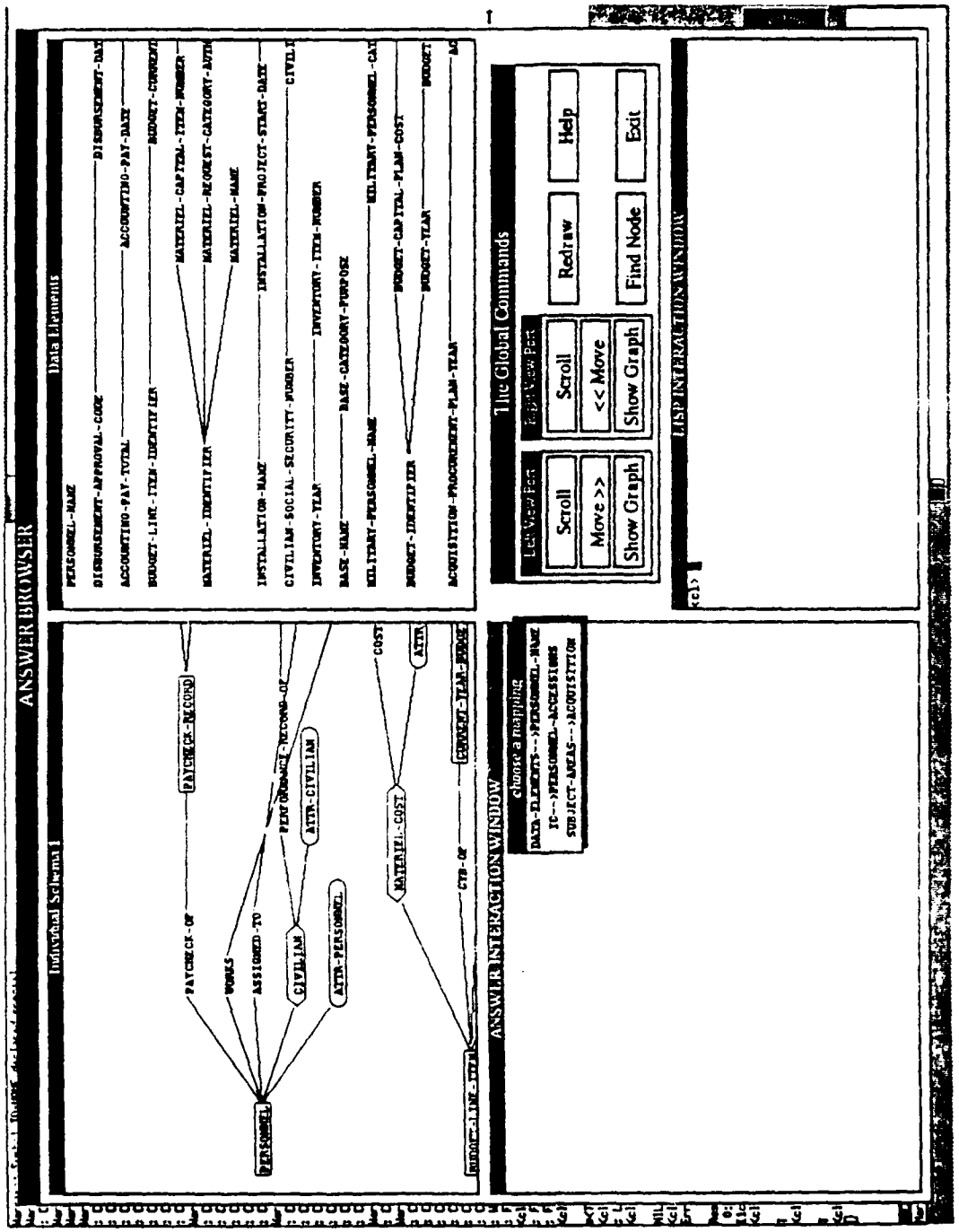


Figure 4-6. Browser Window Displaying Associated Standard Data Elements on One of the Exposed Elements

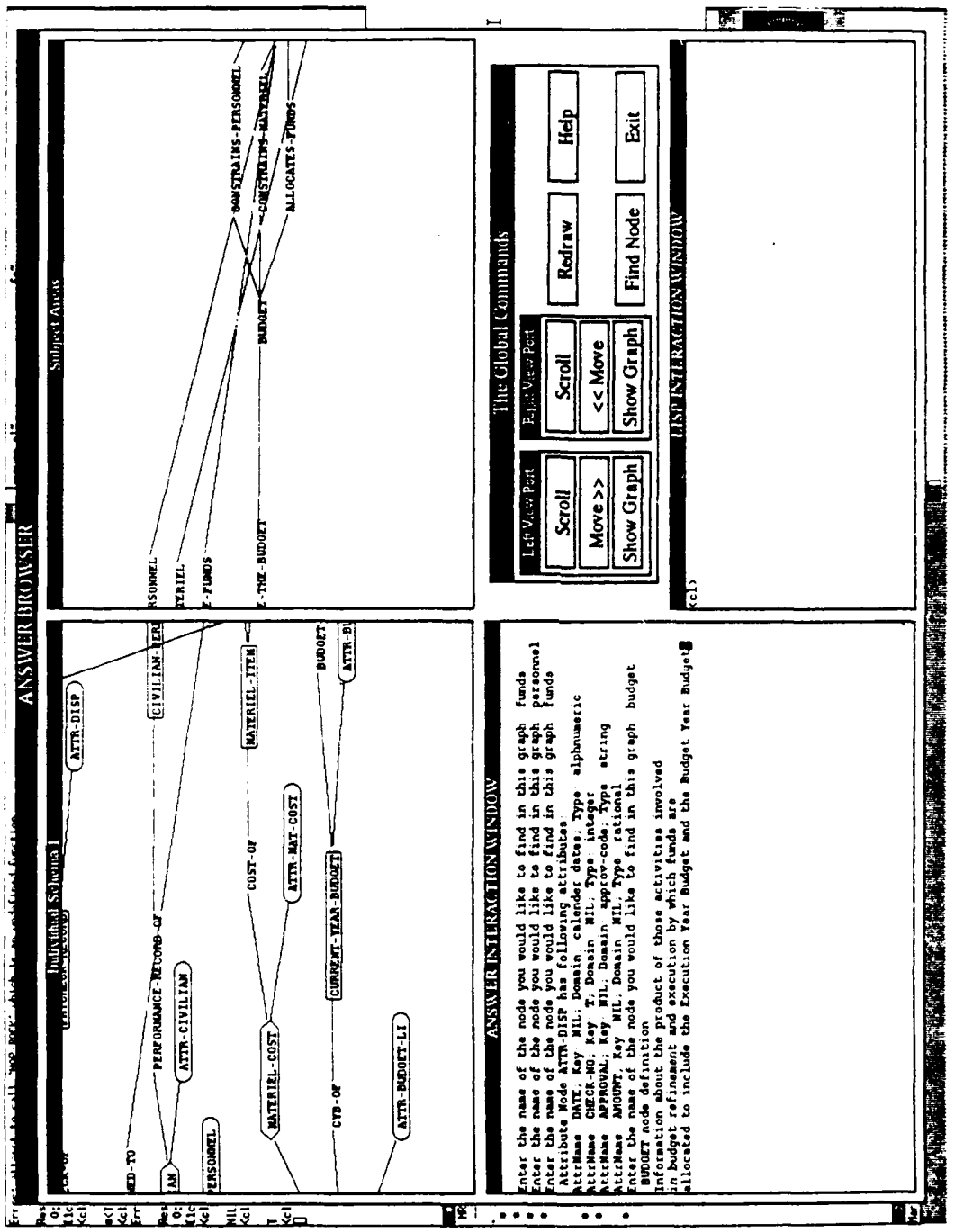


Figure 4-7. Display of Node Definition

Section 5
Applications of Artificial Intelligence to ANSWER

5.1 Introduction

The techniques required to meet the goals of ANSWER depend heavily on sophisticated pattern matching and search techniques. Artificial intelligence has developed a wide range of such techniques. The computational properties of many pattern matching techniques have been very closely studied since many AI applications quickly result in intractable solutions if care is not taken in the implementation. Some of the problems being addressed in ANSWER can also benefit from the pattern matching and search techniques developed in the context of AI applications. As with the AI applications, however, care must be taken in technique application so intractable code is not produced for average case problems.

This section discusses the possible application of certain techniques to schema integration and to browsing tools. The techniques may also be applied elsewhere within the overall ANSWER program. A brief summary of other application areas will be presented at the end of this section.

5.2 Schema Integrator

The basic problem for schema integration is to take two (or more) schemas, collect assertions from a database administrator about the relationships between the schemas, and produce a third integrated schema. The integrated schema reflects the content of the DBA's assertions and the ways in which the original schemas are related. It is a merger of the input schemas where some of the original schema objects may be removed (if equivalent to object in the other schema), renamed, or changed in some other fashion (e.g., an entity in an input schema may become a category in the integrated schema).

The problem can be viewed abstractly as the production of a graph, G[i] from a set of graphs G[a-n] and a set of assertions A[a-n]. The graph G[i] may be produced from G[a-n] and A[a-n] using a variety of operations. The current schema integrator produces G[i] with a large set of procedures and intricate data structures. The size of the code makes it difficult to maintain and modify. The graph G[i] could be produced using other more compact techniques. This may result in an overall improvement in maintainability. The techniques that could be used have been very closely studied and certain formal results are available about their computational properties [SCHM83], [LEVE87], [PATE89], [NEBE88]. Reimplementing the schema integrator drawing on these techniques would result in a tool with a more more solid theoretical grounding. Effects of any modifications to the integration algorithm could be more easily calculated and the code may be easier to maintain. The possible operations include:

- Unification (also sometimes called classification),
- Generalization.

Using these techniques, the solution must be approached as one where some object of graph G[a] is merged into graph G[i], the solution graph.

Unification will produce solution graphs where the object, O[a], to be integrated is established as either a subtype of some existing node in G[i] or as a new independent node (a node with no supertypes) in G[i]. For example, if O[a] is Truck and G[i] already contains an object O[i] called Vehicle, then Truck can be integrated as a subtype of Vehicle. If there is no object O[i] in G[i] that semantically corresponds to a supertype of Truck, then Truck would be directly integrated into G[i] as an independent entity.

The relationships of O[a] must be integrated with the relationships of objects in G[i]. The integration can only be done after the objects in G[a] have all been integrated with the objects of G[i]. This integration corresponds to the lattice merging step in the current schema integrator [ELMA86].

Classification alone is not sufficient to produce a compete solution graph G[i]. A complete solution graph will sometimes require the introduction of a new object, O[n], which represents a generalization of some object O[a] in G[a] and some object O[i] in G[i]. For example, a merger of Military_personnel and Civilian may require the introduction of a new object, Personnel, that has all attributes in common between Military_personnel and Civilian and has Military_personnel and Civilian as categories whose parent is Personnel. The operation of creating a new object in this manner is generalization. Similarly to the classification operation, all relationships in which O[n] participates must also be integrated in the result graph G[i].

Integration of two schemas can be expressed as a combination of a unification problem and a generalization problem. The integration operations correspond to generalization, classification and direct copying (where neither generalization or classification apply). The integration relationships that can be asserted between two objects are:

- Equal,
- Contains,
- Is contained in,
- Maybe integrable,
- Disjoint and not integrable,
- Disjoint and may be integrable.

If two objects O[1] and O[2] are equal, the integration of those objects is either O[1] or O[2]. Similarly the unification of two objects O[1] and O[2] is the substitution of one object for the other in the resulting unified structure.

If O[1] contains O[2], the resulting integration treats O[2] as a subtype of O[1]. In the ECR model this is represented as having O[2] be a category of O[1]. The unification of O[1] with O[2] is the categorization of O[2] as a subtype of O[1]. Similar results apply if O[1] is contained in O[2].

The "maybe integrable" assertion results in O[1] and O[2] both occurring in the final integrated structure as though they were initially asserted to be disjoint. That is both O[1] and O[2] will be copied to the final structure. The result of unifying O[1] with O[2] where O[1] and O[2] are disjoint is the same.

The integration result for two objects asserted to be "disjoint an maybe integrable" depends on whether or not the two objects share any attributes in common. If O[1] and O[2] share at least one attribute, the schema integrator will produce a new object D_O[i], which is represented as a parent object of O[1] and O[2]. O[1] and O[2] are categories in the integrated structure. The generalization of O[1] and O[2] will proceed similarly. If O[1] and O[2] share n attributes, the generalization of O[1] and O[2] can be defined to be O[i] with all shared attributes and O[1] and O[2] as subtypes of O[i].

There are no other possible integration relationships. Similarly, unification or generalization of two objects only produces new classification, generalization, substitution or copying results. Thus the unification or generalization of O[i] and O[j] can be used to represent the integration of O[i] with O[j].

The only remaining issue in applying unification to the problem of schema integration is to determine the best method of incorporating the user specified assertions about relationships between objects into a structure that the unification algorithm can manipulate together with the schema descriptions. One possible approach is to treat the schema integration problem as a parsing problem. Under this view, one schema is integrated with another by applying the user assertions as rules that govern the combination of G[i] with G[j]. In effect, a final integrated structure G[n] is initialized to G[i]. G[i] is treated as a structure to which G[j] will be added by applying rules to subgraphs of G[j] and G[i] and producing a new subgraph for G[n]. There are a number of implementation options for this approach. Schemas and assertions can be represented as complex features (or frame structures) to which existing unification-based parsers can be applied [SCHI86]. The integration rules can be used to encode any generalization effects that may be required (i.e., the introduction of a most specific generalizer of two objects cannot be produced directly by unification). To get around this problem, the integration rules can specify that two objects, O[i] and O[j], which are in a "disjoint but integrable" relationship will produce a new object satisfying their most specific generalizer The integration rule can be applied to the objects, O[i] and O[j] by unifying the description of the rule with the descriptions of the objects. The result of applying the rule will be the introduction of the most specific generalizer into the final result graph, along with the properly categorized versions of O[i] and O[j].

If this basic approach to schema integration is adopted, it would have several advantages. The basic results on the computational properties of subsumption language could be applied to the problem of schema integration. In addition, the integration process could be cast in a much more declarative formalism, thereby simplifying creation and maintenance of new schema integration strategies. Strategies could be reflected in differences in the rule set used by the parser.

5.3 Browser

An effective Browser must supply multiple views for any piece of information the user wishes to browse. This is typically achieved through the use of graphics display capabilities that allow the user to view the object of interest and arbitrary amounts of context in an arbitrary order. For example, in the ANSWER Browser, a user may start browsing at any level with any node. There are no restrictions on the number of related nodes or levels a user can view or the order in which nodes and levels are visited.

A critical component of a Browser is the method used to initially select the first object of interest. In the current ANSWER Browser, the user may select a specific level from a menu of levels or the user may name a specific node if the user knows the specific node name. A more flexible Browser would offer additional techniques to identify a node of interest. Simple methods include menus of schema node names structured according to related subject areas, prime terms or standard data elements. Another alternative is to supply a structured query language where the user identifies nodes satisfying a formal description. For example: "Select all nodes that are directly related to an entity called Military_personnel and an entity called Disbursement." Such a query would be useful if the user knew there was more than one schema with an entity called Disbursement but that the particular schema the user wanted was one satisfying that description. Other more complex alternatives to node indexing include increasingly sophisticated ad hoc query languages. A natural language description of nodes to be browsed would provide the greatest amount of flexibility in node indexing. A medium ground between a formal query language and full natural language capabilities would be a restricted sublanguage. Such a language would seem natural to the user but would be computationally cheaper, tracking only semantic and syntactic distinctions that are relevant to minimal effective support of the indexing process.

Another important component of a Browser is the ability to 'grow' a path between two nodes. This sort of facility is useful in cases where a user wants to focus on the relationship between two objects, O[i] and O[j], but the nodes are only related through a long path of other nodes in the graph. If the graph display is centered on O[i] and the user in interested in seeing a particular relationship between O[i] and O[j], the user must follow the graph until O[j] appears in the screen. If the user is not sure exactly where O[j] lies relative to O[i], several false starts may be needed to identify the desired path direction displaying the relationship between O[i] and O[j].

If there is only one possible path between O[i] and O[j], automating the display of that path is straightforward. If there is more than one path between O[i] and O[j], then the automation is more difficult (relative to the number of possible paths). The same techniques needed to support intelligent indexing could also be used to support intelligent path growing.

5.4 Other Application Areas

Other application areas within the ANSWER program may also benefit from the application of AI techniques. In particular, the following areas could derive direct benefit:

- Query formulation,
- Data element standardization efforts.

Query formulation facilities could use the same techniques that may be applied to intelligent indexing for the Browser.

Data element standardization efforts can benefit from AI techniques to create new standard data elements and to identify when a proposed standard data element is redundant with some existing data element. This latter function can be achieved only if an extensive representation of the intended semantic interpretation of each data element is formally modeled. This is quite possible to do using existing knowledge representation and natural language understanding techniques. A redundant data element identifier could apply the same pattern matching techniques discussed for the schema integrator together with natural language semantic interpretation techniques to identify (relative to a fixed semantic interpretation) whether or not the semantic equivalent of a new proposed data element already existed in some other synonymous form. Similar techniques could also be applied for homonym detection, that is, to determine if two associations of a single data element in different schemas resulted in two semantically different uses of the term.

Section 6
Conclusions and Plans

During Phase I of the ANSWER program we have successfully completed the development of two major software deliverables envisioned for the ANSWER architecture. Those deliverables are the encyclopedia manager and the Browser. These software prototypes can be used to help to meet the Army's long-range goals of integrated data management and access of the Army's information systems. The major tasks accomplished as part of Phase I include:

- ***Requirements analysis***—Definition of the long-term goals of ANSWER and specific requirements for the short term.
- ***System design***—Definition of the information and functional architectures for the ANSWER system.
- ***Representation model***—Selection of a representation model to describe ANSWER information. The entity-category-relationship (ECR) model was selected.
- ***Encyclopedia manager***—Implementation of a schema integrator tool, implementation of IRDS running with ORACLE in a UNIX environment, implementation of tools to support Army data dictionary activities.
- ***Browser***—A graphical Browser prototype has been implemented. The Browser can display all structures of the ANSWER information architecture.

Phase II of the ANSWER program will track the installation and use of the Browser and the encyclopedia manager at an Army installation. Modifications and revisions may be made to the prototypes as indicated by the experiences of the target user group. Phase II also includes additional development of new tools to aid in the support of ANSWER goals. The Phase II tasks are:

- ***Database registration automation***—In this task we will design modifications to the schema integrator tool that reflect the results of use of the tool by a target group of users. Selected modifications will be implemented for demonstration at the end of Phase II.
- ***AI techniques implementation***—This task will investigate database registration tools beyond the schema integrator that are applicable to the problem of database registration. Such tools might include data element standardization tools, flat file system conversion tools and possibly others. Possibilities for such tools will be investigated in Phase II in cooperation with Army agencies identified with AIRMICS assistance.

- ***Demonstration and training.***

- ***User interface***—In this task we will implement an X-Windows-based user interface that allows the user to invoke the ANSWER tools, including the schema integrator tool, the Browser, IRDS and other tools to be defined as part of the AI techniques task.

- ***Browsing***—In this task we will design possible modification to the browsing tool developed and delivered as part of Phase I. The modifications will be based on feedback received from a target user group using the software deliverables from Phase I.

Phases III and IV will continue the efforts and move the ANSWER program towards a complete implementation of the long-range architecture goals.

The tasks for Phase III include:

- ***Implementing Browser enhancements***—In this task we will implement enhancements to the Browser identified in Phase II. We will also investigate issues associated with implementing the Browser on a large realistic model.

- ***Query formulation implementation***—In this task we will analyze approaches for query formulation, make recommendations for implementation and review these with the ARMY representatives to select appropriate approaches. Approaches to be analyzed include syntax diagram display, SQL syntax error detection, draft SQL execution, intelligent user interfaces, automatic formulation of SQL requests from Browser examples, natural language paraphrases of SQL requests and natural language queries translated into SQL. We will design and implement (as necessary) query formulation tools that are selected.

- ***AI techniques implementation***—This task is a continuation of the new Phase II task for identifying additional tools for database registration. In this task we will identify the most useful tools and produce prototype implementations.

- ***Security study***—In this task we will develop an overall approach for security enforcement in ANSWER, and make recommendations for implementation and exploration of key concepts.

- ***Demonstration and Training.***

- ***Distributed query processing***—In this task we will identify existing commercially available distributed query processing capabilities and design the integration of ANSWER tools with the distributed query processing capability.

Phase IV represents the remaining six months of the original Phase III of the contract. The major task in Phase IV is the implementation and integration of the ANSWER tools with existing Army distributed query processing facilities. The scope of these tasks will be discussed with the Army in 1990. The tasks include:

- ***Distributed query processing***—This task will implement the integration of ANSWER tools with a distributed query processing facility.

- ***Demonstration and training***—Install the ANSWER system on hardware at an ARMY installation. Conduct training sessions in the installation, use and maintenance of the ANSWER system.

We are confident that the ANSWER program will continue to successfully meet its goals. We intend to continue actively working with Army personnel to identify tools of greatest benefit to achieving the long-range goals of integrated data access and management for heterogeneous computer systems. The feedback we receive from the delivered prototypes in Phase I will enable us to identify and develop additional effective software tools for the Army's data management goals.

Section 7
References

[ACMP80] Proceedings of the Workshop on Data Abstraction, Databases, and Conceptual Modeling, *SIGMOD Record*, Vol. 11, No. 2, February 1981.

[CHEN76] Chen, P.P.S., "The Entity-Relationship Model: Toward a Unified View of Data," *ACM Transactions on Database Systems*, Vol. 1, No. 1, March 1976.

[ELMA85] El Masri, R., Hevner, A. and Weeldreyer, J."The Category Concept: An Extension to the Entity-Relationship Model," *Data and Knowledge Engineering Journal*, Vol. 1, No. 1, 1985, pp. 75-116.

[ELMA86] Ramez El Masri, James A. Larson, Shamkant Navathe, "Schema Integration Algorithms for federated Databases and Logical Database Design," CSDD Technical Report, CSC-86-9:8212, January 1986.

[HULL87] Hull, Richard, and Roger King, "Semantic Database Modeling: Survey, Applications, and Research Issues," *ACM Computing Surveys*, Vol. 19, No. 3, September 1987, pp. 201-261.

[KENT84] Kent, Wm., "Fact-Based Data Analysis and Design," *The Journal of Systems and Software*, Vol. 4, pp. 99-121, 1984.

[LEVE87] Levesque, H., and Brachman, R. "Expressiveness and Tractability in Knowledge Representation and Reasoning," *Computational Intelligence* 3(2), 1987, pp. 78-83.

[MART84] Marti, Robert W., "Integrating Database and Program Descriptions Using an ER Data Dictionary," *The Journal of Systems and Software*, Vol. 4, pp. 185-195, 1984.

[NBSIR 88-3700] Goldfine, Alan, and Patricia Konig, "A Technical Overview of the Information Resource Dictionary System," NBSIR 88-3700, January 1988.

[NEBE88] Nebel, B., "Computational Complexity of Terminological Reasoning in BACK," *Artificial Intelligence*, Vol. 34, 1988, pp. 371-383.

[PATE89] Patel-Schneider, Peter, "A Four-Valued Semantic for Terminological Logics," *Artificial Intelligence*, Vol. 38, 1989, pp. 319-351.

[SCHI86] Schieber, Stuart, "An Introduction to Unification Based Approaches to Grammar," Tutorial presented at 23rd Annual Meeting of the Association for Computational Linguistics, July 8, 1985, Chicago, Illinois.

[SCHM83] Schmolze, J.G., and Israel, D.J., KL-ONE: Semantics and Classification," Technical Report 5421, BBN Laboratories, 1983. Part of a collection entitled *Research in Knowledge Representation for Natural Language Understanding - Annual Report*, September 1, 1982 - August 31, 1983.